# AN INVESTIGATION OF SELECTION METHODS FOR A SIMPLE PROGRAM FLOW ANALYSIS ALGORITHM

Norbert Lukasczyk

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93940

# NAVAL POSTGRADUATE SCHOOL
## Monterey, California

# THESIS

AN INVESTIGATION OF SELECTION METHODS
FOR A SIMPLE
PROGRAM FLOW ANALYSIS ALGORITHM

by

Norbert Lukasczyk

June 1974

Thesis Advisor: G. A. Kildall

Approved for public release; distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) An Investigation of Selection Methods for a Simple Program Flow Analysis Algorithm | | 5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; June 1974 |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) Norbert Lukasczyk | | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940 | | 12. REPORT DATE June 1974 |
| | | 13. NUMBER OF PAGES 85 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940 | | 15. SECURITY CLASS. (of this report) Unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

code optimization
flow graph analysis algorithm
investigation of selection methods in a simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The problem of code optimization during compilation can be approached in different ways. Kildall(6) conducted an analysis of the program structure to produce optimized object code during compilation. He utilized a directed graph to represent the program flow, along with an "optimizing function," an "optimizing pool," and a "meet operation." Based on these concepts, his program flow analysis algorithm collected corresponding graph

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
(Page 1) S/N 0102-014-6601 |

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.

elements on an "investigation list" and processed then those elements one at a time. The algorithm, as presented, does not specify a sequence in which these elements are selected from the list. This thesis investigates four selection methods: "Last In First Out," "First In First Out," "Steepest Descent," and "Depth First Search," a method developed by Ullman (3). The convergence rate of the methods is evaluated by comparing the number of applications of the meet and optimizing operations in a simulated optimizing environment.

An Investigation of Selection Methods
for a simple
Program Flow Analysis Algorithm

by

Norbert Lukasczyk
Lieutenant Commander, Federal German Navy

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the
NAVAL POSTGRADUATE SCHOOL
June 1974

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93940

# ABSTRACT

The problem of code optimization during compilation can be approached in different ways. Kildall[6] conducted an analysis of the program structure to produce optimized object code during compilation. He utilized a directed graph to represent the program flow, along with an "optimizing function," an "optimizing pool," and a "meet operation." Based on these concepts, his program flow analysis algorithm collected corresponding graph elements on an "investigation list" and processed then those elements one at a time. The algorithm, as presented, does not specify a sequence in which these elements are selected from the list. This thesis investigates four selection methods: "Last In First Out," "First In First Out," "Steepest Descent," and "Depth First Search," a method developed by Ullman[3]. The convergence rate of the methods is evaluated by comparing the number of applications of the meet and optimizing operations in a simulated optimizing environment.

TABLE OF CONTENTS

## LIST OF FIGURES



## LIST OF TABLES

# I. INTRODUCTION

A particular problem with mechanical translation of high level languages into machine code is that more efficient code can often be written by an assembly level programmer, where efficiency is measured in the executior time for the program and the amount of memory required. The necessity to overcome this shortcoming created the science of code optimization.

The goal of code optimization is to modify the representaticn of a given program P intc a machine code presenting a program P' such that the program P' will perform the same function as program P, but is more efficient, using less execution time, smaller storage, or fewer registers.

One global optimizing technique called interval analysis [2] partitions the flow graph into subgraphs called intervals. Each interval then is replaced by a single node containing the local optimizing information. The resulting graph is partitioned into subgraphs again, and the corresponding intervals are replaced by nodes. The definiticn of such interval partitions is done until the graph becomes a single node itself, at which time global information is propagated locally by reversing the reduction process.

Hecht and Ullman [2] presented a simple "bit propagation algorithm" which uses a special ordering of nodes (explained in section III-D) for a reducible graph, where a reducible graph is one which can be reduced by the interval approach from the initial flow graph to the limit flow graph of a single node with no edges. In their approach each node has an associated bit vector containing the optimizing data. This bit vector is updated during the reduction of the graph to the final solution.

Another approach is done by Kildall [6]. He conducts the analysis using a program flow analysis algorithm, which propagates information along the program flow until all required information is collected. This algorithm is reviewed in section III-B. The algorithm, as presented, does not specify the sequence in which the basic blocks are processed. This thesis will investigate some selection methods which specify the sequences in which the basic blocks are processed. These selection methods are simulated and compared with respect to the convergence rate of the algorithm.

# II. BACKGROUND

## A. DEFINITIONS

Before reviewing the algorithm[6] it is necessary to define the following terms.

A directed graph is a two tuple GD = (N,A), where "N" is a finite set of nodes and "A" is a subset of N x N, called "arcs." The arc (X,Y) leaves node X and enters node Y. Node X is called a predecessor of node Y and node Y is called a successor of node X.

A path from node B to node C in GD, for B,C in N is a sequence of nodes $(X_1,X_2,....X_k)$ such that $X_1 = B$ and $X_k = C$ and $(X_i,X_{i+1})$ is an arc in A for all i, $1 \le i \le k-1$ .

The length of a path is one less than the number of nodes in the sequence (k-1 in the above case). If C equals B the path is called a cycle.

A program flow graph is a triple G = (N,A,E) where N and A are as defined above and where E is a subset of N and contains the "entry node(s)," such that given a node n in N there exists a path $(X_1,X_2,.....X_k)$ where $X_1$ is in E and $X_k$ equals n.

An optimizing pool associated with each node in the graph is a set describing the optimizing information associated with the particular node in terms of the analysis being conducted. For example this set may contain subexpressions as elements, register allocation information, or propagated constant values.

An input pool is the set of optimizing information elements entering a node.

An output pool is the set of optimizing information

elements leaving a node.

A <u>meet operation</u> is defined which combines two or more pools at a node, where two or more program paths join. The form of the meet operation varies with the type of analysis. Formally, the meet operation, denoted by "$\wedge$", is a binary operator which maps P x P into P, where P is the set of all optimizing pools. The meet operator has the following properties:

For a,b,c in P

$a \wedge a = a$ (idempotent)

$a \wedge b = b \wedge a$ (commutative)

$a \wedge (b \wedge c) = (a \wedge b) \wedge c$ (associative)

The meet operation permits the definition of a partial ordering on the optimizing elements :

$a \geq b$ iff $a \wedge b = b$.

To simplify the notation, the form

$\bigwedge_{1 \leq i \leq k} X$ is defined as $x_1 \wedge x_2 \ldots\ldots \wedge x_k$.

An <u>optimizing function</u> f maps a given input optimizing pool to the output pool of the corresponding node which contributes to the input pools of the node's immediate successors. The function differs with the type of analysis being conducted. The function, however, must satisfy the following homomorphism condition for Kildall's data flow analysis algorithm to be applicable :

$f(n,a \wedge b) = f(n,a) \wedge f(n,b)$

for all nodes n in N, and a,b in P.

<u>A zero element 0</u> is an element of P satisfying the conditions

$0 \wedge X = 0$ for all X in P.

<u>A one element 1</u> is an element of P satisfying the condition

$\underline{1} \wedge X = X$ for all X in P .

B. REVIEW OF KILDALL'S ALGORITHM

Kildall's program flow analysis algorithm is used in order to perform compile time optimization of object code. Although the algorithm is optimization independent several different useful optimization functions have been applied, such as locating redundant computations or register operations. In the following paragraphs this algorithm will be reviewed, and later will be illustrated by an example of common subexpression analysis. The algorithm can be described in the following way :

<u>STEP 1</u>: Initialize an investigation list "L" by the entry nodes of the program graph along with the corresponding optimizing pools "OP." Normally there is only one entry node and its optimizing pool is empty (initialized to the zero element ).

<u>STEP 2</u>: If the list "L" is empty then halt. All nodes of the graph have been processed (at least once) and the optimizing pools are in their final state.

<u>STEP 3</u>: Otherwise, select a node "X" from "L" with its corresponding (already established) optimizing pool. When the node is processed the first time, assume the approximate pool to be initialized to the $\underline{1}$ element.

<u>STEP 4</u>: Use the meet operator to combine the already existing optimizing pool with the input pool "IP" incoming from the immediate predecessor. Assume the result as the new optimizing pool. If the result does not change the existing optimizing pool go to "STEP 2."

<u>STEP 5</u>: Otherwise map the result to a new output pool with the corresponding optimizing function. Enter all immediate successors into "L" with the output pool of X as a new in put pool.

<u>STEP 6</u>: Go to STEP 2. In general the algorithm is stated as

A1 [Initialize ] $L \leftarrow \{(e,\underline{0}) \mid e \text{ in } E\}$

A2 [Terminate ?] If $L = \emptyset$ then halt

A3 [Select a node] Let L' be an element of L,
L'=(X,IP) fcr some X in N
and IP in P, where P is the set of all
possible optimizing pcols, then set
$L = L - \{L'\}$

A4 [Traverse?] Let OP be the current pool
of optimizing informaticn associated
with the node X ( initially OP= $\underline{1}$ )
If OP $\leq$ IP go to STEP A2,where IP is
the incoming pool.

A5 [Set pool ] $OP \leftarrow OP \wedge IP$
$L = L \cup \{(Y_i , f(X,OP)) \mid Y_i \text{ in } I(X),$
where I(X) is the set of all
immediate successors of X. }

A6 [Loop ] Go to STEP A2

As an illustration of the algorithm, consider the problem cf common subexpression elimination. In this case the "pools" are sets partitioned into equivalence classes. Such equivalence classes contain previously computed expressions which are known to have identical values. The meet operation is defined as intersection of equivalence

classes, and the "optimizing function" builds new equivalence classes of expressions which are known to have the same value, or it adds expressions to already existing classes. As an example consider the following skeletal program:

```
    z := y
1 : r := k * y
    x := k
    if c = 1 go to 2
    y :=z
    r := u * z
    x := u
    if 1 ≤ c go to 1
2 : u * z
    x * y
```

Neglecting the if statements, the resulting program flow graph is presented in Figure 1, where the nodes A,B,C,D,E,F represent basic blocks which are segments of the program containig no transfers of the program control into or out of the segment and where the edges represent the program flow.

The program flow graph given in Figure 1 is processed in the following tabular form :

Column "STEP" contains the number of nodes already processed.

Column "NODE" contains the node being processed.

Column "INPUT POOL" contains the current approximation to the final optimizing pool.

Column "OUTPUT POOL" contains the output pool formed by the optimizing function which will be the input pool for the immediate successor listed in column L.

The single equivalence classes are separated by a vertical bar, and the single elements of a class are separated by a comma. The analysis is given in Table 1. The optimizing pools associated with the underlined nodes are the final pools.

The result of this analysis is as follows. The first

expression at node D is Y := Z. Referring to the input pool

$$IP_{(D)} = \{z,y|k,x,|r,k*y,x*y\}$$

there already exists an equivalence class "|z,y|" and therefore it would be redundant to produce code again for this assignment. A similar result is found by the algorithm at node F. The expression X * Y is already a member of the input pool

$$IP_{(F)} = \{z,y|x|u|k|r,x*y|u*z|k*y\}$$

and thus it is not necessary to recompute the expression.

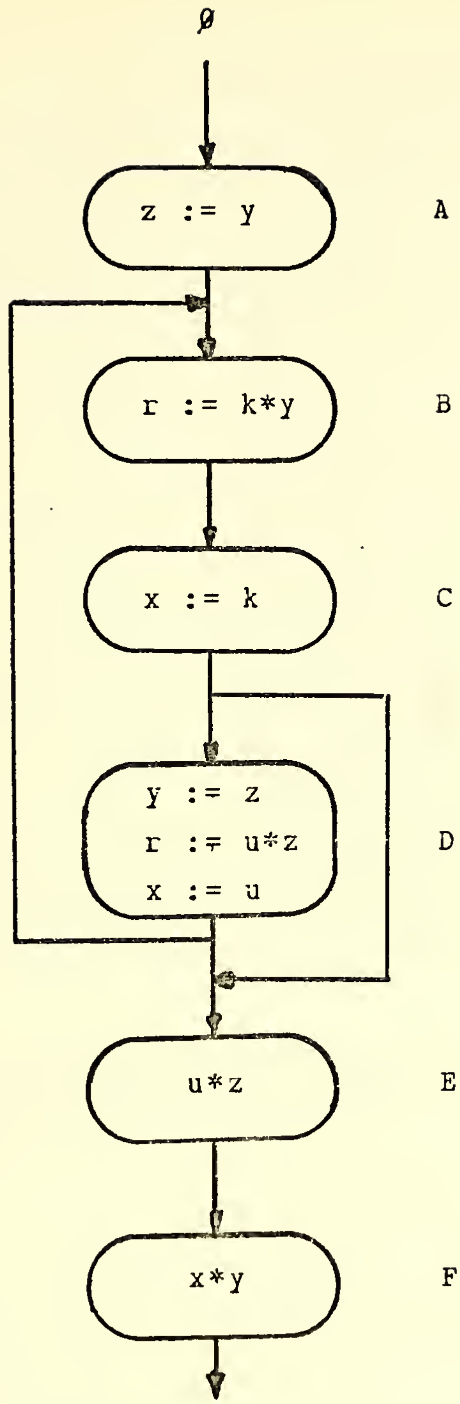


FIGURE 1.

Example for a program flow graph

TABLE 1

Common Subexpression Elimination Analysis

<table>
<tr><th>STEP</th><th>NODE</th><th>INPUT POOL</th><th>OUTPUT POOL</th><th>L</th></tr>
<tr><td>1</td><td></td><td></td><td>∅</td><td>A</td></tr>
<tr><td>2</td><td>A</td><td>∅</td><td>z,y</td><td>B</td></tr>
<tr><td>3</td><td>B</td><td>z,y</td><td>z,y|r,k*y,k*z|k</td><td>C</td></tr>
<tr><td>4</td><td>C</td><td>z,y|r,k*y|k</td><td>z,y|k,x|r,k*x,x*y, x*z</td><td>D,E</td></tr>
<tr><td>5</td><td>E</td><td>z,y|k,x|r,k*y,x*y</td><td>z,y|k,x|u|r,k*y, x*y|u*z</td><td>F</td></tr>
<tr><td>6</td><td>F</td><td>z,y|k,x|u|r,k*y, x*y|u*z</td><td>z,y|k,x|u|r,k*y, x*y|u*z</td><td>-</td></tr>
<tr><td>7</td><td>D</td><td>z,y|k,x|r,k*y,x*y</td><td>z,y|x,u|r,u*z, x*y|k*y|k</td><td>E,B</td></tr>
<tr><td>8</td><td>E</td><td>z,y|x|r,x*y|k*y|k</td><td>z,y|x|u|k|r,x*y| u*z|k*y</td><td>F</td></tr>
<tr><td>9</td><td>F</td><td>z,y|x|u|k|r,x*y, u*z|k*y</td><td>z,y|x|u|k|r,x*y| u*z|k*y</td><td>-</td></tr>
</table>

Note: cnly expressions or subexpression which cccur in the graph are propagated.

## III. SELECTION METHODS

The choice of the node from the investigation list in step 3 of Kildall's algorithm is arbitrary, as shown in Ref[6]. This fact has motivated the investigation of the effect of some selection methods on the convergence rate of the algorithm.

In the analysis of the previous example of common subexpression elimination presented in Table 1, there was a choice of nodes at step 4. Instead of node E the node D could have been taken. Proceeding with the choice D instead of E reduces the number of steps leading to the final solution from nine to seven, as shown in Table 2.

### A. LAST IN FIRST OUT (LIFO)

Representing the investigation list in form of a horizontal stack suggests a selection method which always processes the top of the stack, assumed to be the "right end." Due to step 5 of the algorithm, the currently processed node enters all its immediate successors at the stack top. The actions of the algorithm are most easily seen through a simple example. Considering the skeletal flow graph given in Figure 2, stack order processing produces the following states of column L. Processing the entry node A results in:

$$L_1 = B\ C\ D.$$

Processing node D results in

$$L_2 = B\ C\ I\ J.$$

Selecting J produces

$$L_3 = B\ C\ I\ L\ M.$$

TABLE 2

Common Subexpression Elimination Analysis

Improved version

| STEP | NODE | INPUT POOL | OUTPUT POOL | L |
|---|---|---|---|---|
| 1 | | | ∅ | A |
| 2 | A | ∅ | z,y, | B |
| 3 | B | z,y | z,y\|r,k*y,k*z\|k | C |
| 4 | C | z,y\|r,k*y\|k | z,y\|k,x\|r,k*x,x*y, x*z | D,E |
| 5 | D | z,y\|k,x\|r,k*y,x*y | z,y\|x,u\|r,u*z, x*y\|k*y\|k | E,B |
| 6 | E | z,y\|x\|r,x*y\|k*y\|k | z,y\|x\|u\|k\|r,x*y\| u*z\|k*y | F |
| 7 | F | z,y\|x\|u\|k\|r,x*y, u*z\|k*y | z,y\|x\|u\|k\|r,x*y\| u*z\|k*y | – |

Note: only expressions or subexpression which occur in the graph are propagated.

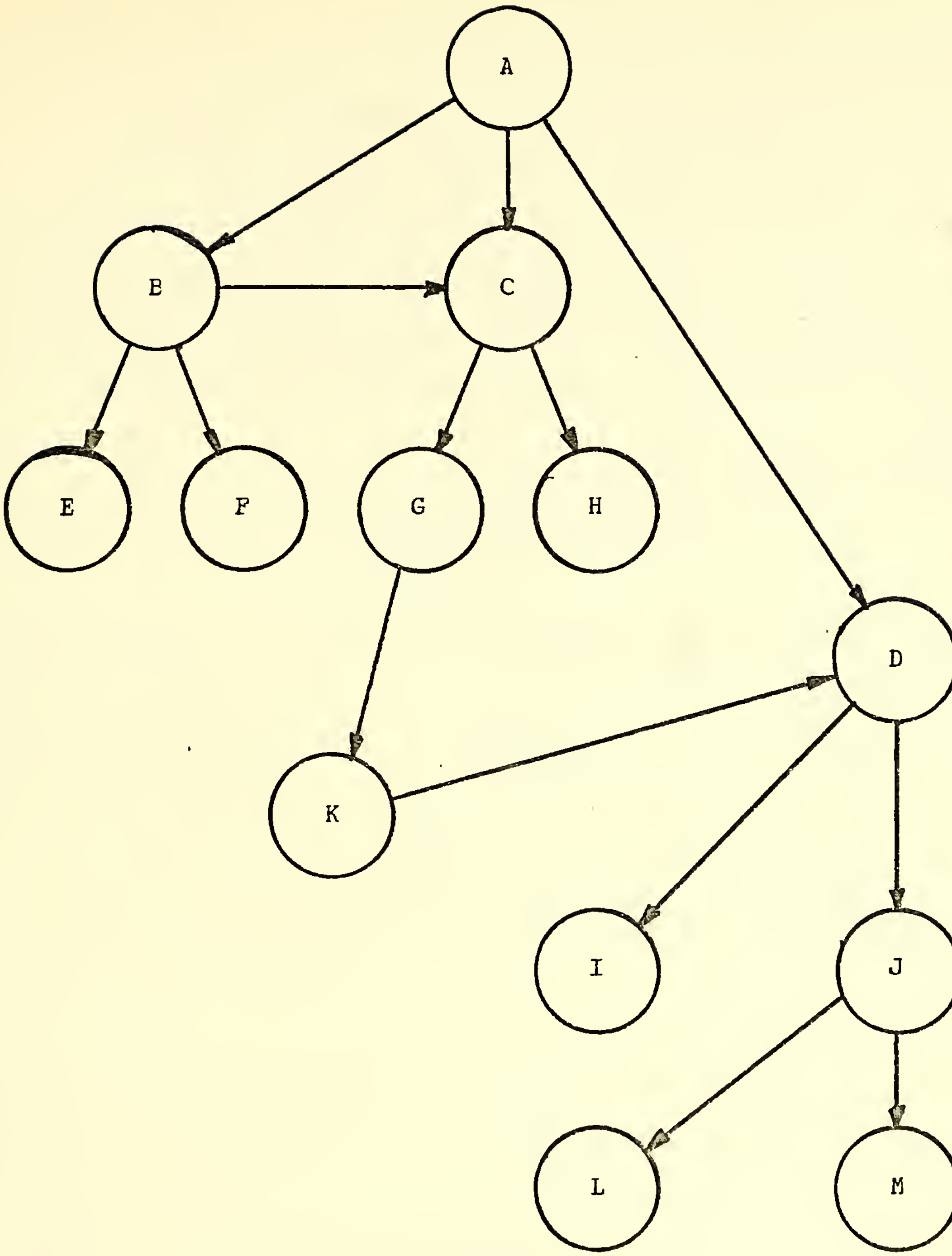


FIGURE 2.
Flow graph example for
LIFO and FIFO node selection

Since node L and M have no successors, the subtree rooted in D is processed. The optimizing pools (OP) for the nodes I, J, L, M are of the form:

$$OP_{(n)} = \bigwedge_{D->n} f(OP_D) \text{ for } n = I,J,L,M$$

where $OP_D$ is the result produced by the meet operation along the path from node A to node D, and D->n is a path from D to n. The next node being processed is node C, and therefore the list will be of the form

$$L_4 = B\ G\ H.$$

Since H has no successors the node G is processed leading to K which, in turn, is immediately processed.

$$L_5 = B\ K.$$

Assuming that the output pool of node K will change the $OP_D$, the whole subtree rooted in D has to be reentered and processed again as shown in $L_1$ to $L_3$. The optimizing pools of the nodes G, H, D, I, J, L, M are then of the form:

$$OP_{(n)} = \bigwedge_{C->n} f(OP_C) \text{ for } n = G,H,D,I,J,L,M.$$

After reprocessing the nodes, B is processed which results in the list:

$$L_9 = E\ F\ C.$$

Assuming again that the output pool of B changes $OP_C$, the subtrees rooted in C and D have to be reentered and processed again.

$$L_{10} = E\ F\ G\ H$$

$$L_{11} = E\ F\ K$$

$$L_{12} = E\ F\ D$$

$$L_{13} = E\ F\ I\ J$$

$$L_{14} = E\ F\ I\ L\ M.$$

After doing so, the optimizing pools for the nodes represented by n are:

$$OP_{(n)} = \bigwedge_{B->n} f(OP_B)$$

for n = E,F,C,G,H,K,D,I,J,L,M.

Since node E and node F have no successors, the stack is empty after processing each element and the algorithm stops.

An obvious disadvantage was to start at the right side of the stack, but the graph given in Figure 2 is, of course, only one of many. There might be graphs for which this method is more efficient. For example, if one turns the given graph from left to right the entering sequence will then be D C B. In this case, the algorithm converges more rapidly.

To describe the relations between the nodes by more associative names, call the successors "sons," the sons of a father "brothers," and consider the immediate successors of corresponding brothers to be on one "level." For example, the node A is the father of B C D and the nodes E F G H I J are on one level. The selection method described above propagated the optimizing information along a path connecting sons of succeeding levels in "top to the bottom," or "depth first" fashion. The disadvantage is that all optimizing information entering nodes on this path, coming from nodes of upper levels but being located at the left side of the stack, is disregarded. Assuming that the incoming pool will change the optimizing pool of the entered node, the whole path following the entered node has to be processed again, as illustrated in the case of the subtree rooted in D when node K or B was processed.

## B. FIRST IN FIRST OUT (FIFO)

Based upon the analysis presented in the last section the method presented here selects the nodes in a horizontal direction instead of a depth first sequence. That is at

first all nodes of one level are processed, followed by their successors, again all those of one level, and so on.

Considering the same stack model, the nodes are taken now from the left side, and the node which enters first will also be processed first. Processing the graph of Figure 2 results in the following investigation lists. The node A will produce the list

$$L_1 = B\ C\ D.$$

B will then be selected. Processing the node B will enlarge the list to

$$L_2 = C\ D\ E\ F\ C.$$

The node C is entered twice into the list and is distinguished by the associated optimizing pools. Node A caused its successor C with the approximate optimizing pool

$$OP_C = \wedge f(OP_A).$$

Node B on the other hand, enters the successor C with the pool

$$OP_C = \wedge\ f(OP_B).$$

The following change is made to improve the algorithm in later steps. Instead of adding the corresponding successors in step 5 of the algorithm, the "union" is formed of the existing list L and the set of successors. For those nodes already on the list, the meet operator is applied to combine the different incoming pools.

It is possible to use the union operation since the character of the investigation list is one of a "waiting list" indicating those nodes which remain to be processed. It is easily shown that as long as the input pool is updated by the meet operation it is not necessary to enter a node again. Using this change in step 5 reduces the number of optimizing function applications by the number of times the node is not additionally entered.

Processing the algorithm in this way using the union operation results in

$$L_3 = D\ E\ F\ G\ H$$

$$L_4 = E\ F\ G\ H\ I\ J$$

$$L_5 = H\ I\ J\ K$$

$$L_6 = K\ L\ M$$

$$L_7 = L\ M\ D$$

$$L_8 = I\ J$$

$$L_9 = L\ M.$$

After processing these nodes the list is empty and the algorithm stcps.

Assuming the same relation of the optimizing pocls to one other as in section III-A one improvement can already be noticed: the nodes M and L are processed only twice.

C. STEEPEST DESCENT

The two methods presented in the previcus section only considered the sequence of the nodes based upon the relationship father, son, or brother. The third method presented now will take into account the properties of the associated optimizing pool of each node waiting cn the investigation list. Kildall[6] suggestes a method called "steepest descent" which considers the size of the pccl in obtaining a selection sequence of nodes with the intention of decreasing the pool size as quickly as possible.

The general character of the meet operatcr is similar to that of an intersection operator: that is, in the case of common subexpression eliminaticn, it maps the incoming pools intc a new optimizing pool for the corresponding node. This implies that the size of the produced optimizing pool is always less than or equal to the former one. Therefore, processing the node with the smallest pool available on the investigation list first will map the correspcnding

successors' pools into a form which is closer to the final state.

To illustrate this method consider the shortest path problem (Ref.[1]). In the graph given in Figure 3, each node has a distance associated with it represented by a number. The approximated optimizing pool is the sum of distances along a traveled path beginning at the entry node up to the currently processed one. The meet operator will choose the smallest incoming value as the new optimizing pool. The optimizing function will add the associated distance of the node to its optimizing pool to form the output pool which is passed to the successors. The term "smallest" in this example is related to the sum represented by the optimizing pool.

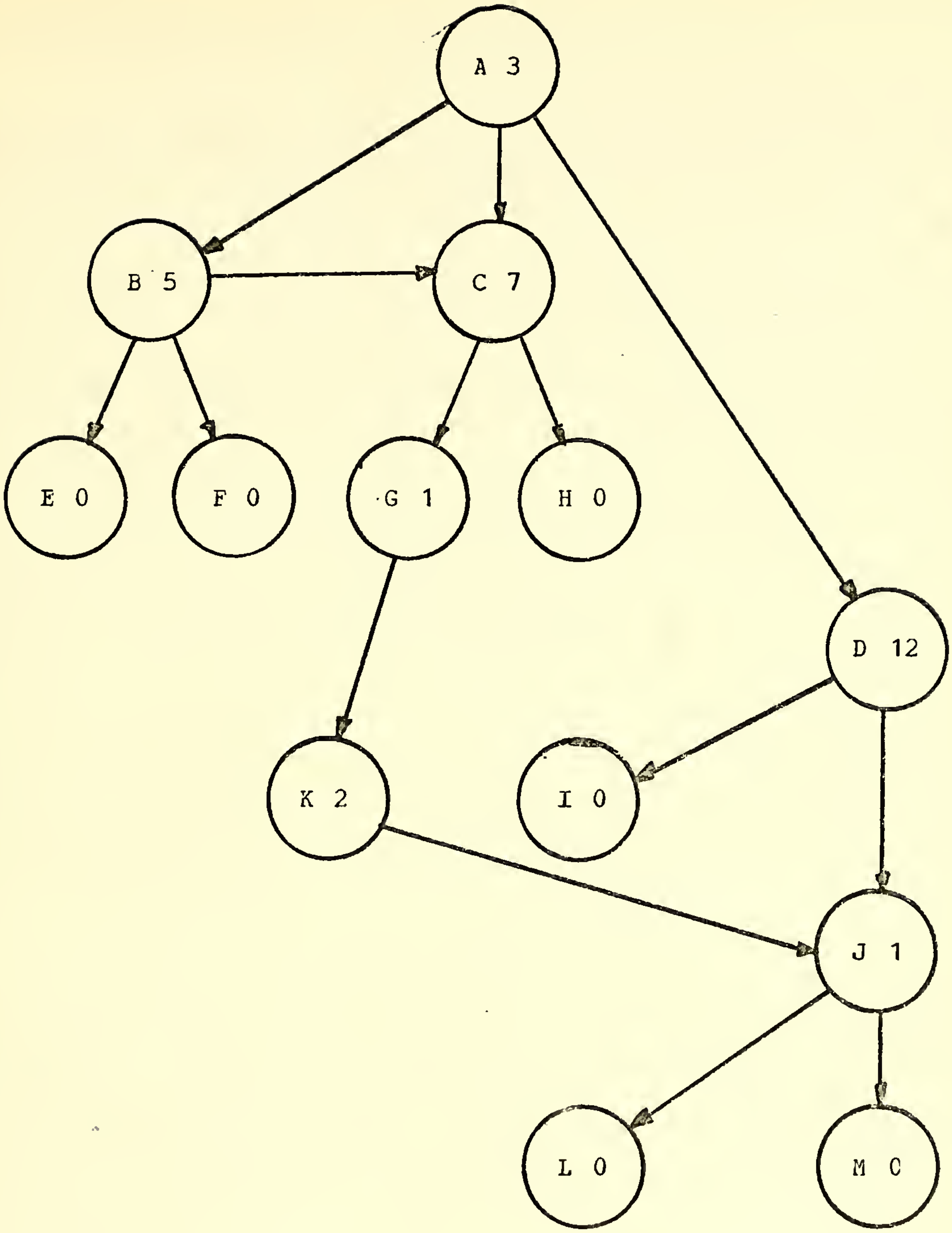


FIGURE 3.
Flow Graph Example for
Steepest descent

Processing the node A will produce the list

$$L_1 = B(3), C(3), D(3).$$

Since all optimizing pools are equal node B is arbitrarily processed next, and the new list is

$$L_2 = C(3), D(3), E(8), F(8).$$

Selecting C(3)

$$L_3 = D(3), E(8), F(8), G(10), H(10).$$

Selecting D(3)

$$L_4 = E(8), F(8), G(10), H(10), I(15), J(15).$$

Selecting E(8), F(8), G(10)

$$L_5 = H(10), I(15), J(15), K(11).$$

Selecting H(10), then K(11) resulting in

$$L_6 = I(15), J(13).$$

Note that the optimizing pool at node J had the value 15 for an optimizing pool. The incoming pool was of value 13 and, due to the meet operator the smaller value was taken as the new optimizing pool. Applying the steepest descent method, node J is taken and

$$L_7 = I(15), L(14), M(14).$$

According to this method, the sequence of selection is L(14), M(14), I(15). Compared to the methods presented previously, the nodes M and L are processed only once which is a further improvement. The change in the graph does not influence the number of traversals of node L and node M.

## D. DEPTH FIRST SEARCH

One consequence of the methods presented in section III-A and III-B was the importance of the processing sequence of the nodes along the possible paths. The relations father - son and brother - brother were considered in determining the best sequence. Another approach will be to preprocess the graph in order to find the relations between all nodes of the graph to each other in the sense of a family tree. The nodes will then be selected in a sequence conforming to the family hierarchy.

This idea has been implemented by Hecht and Ullman[2]. Kam and Ullman[3] combined this ordering process with Kildall's flow graph algorithm. This combination will be presented as the fourth selection method.

The ordering of the nodes of a flow graph G corresponds to the dominance relation and is the reverse of the order in which a node is last visited while growing any depth first spanning tree of G.

A depth first spanning tree (DFST) of a flow graph is defined as a directed rooted ordered spanning tree grown by the algorithm which is described below:

[D1 Initialization ] The root of the DFST is the initial node of G. Let this node be the node "m" which is visited in step D2. The variable "i" will be used to number the nodes in "rEndorder." Initially i<-K, where K is the number of nodes in the graph.

[D2 Visit node m] If node m has a successor X not already on the DFST, select X as the right most son of m found so far on the spanning tree. If this step is successful, node X becomes the node m to be visited next by repeating step D2, otherwise

[D3 Label ] Let m be the node being visited. Let rEndorder <- i , and i<-i-1, if m is the root then halt, otherwise execute step D2 using the father of m. An example of this algorithm is illustrated in Figure 4.

The optimizing pools of the corresponding nodes are processed then in a sequence determined in the following way.

Kam and Ullmann use an array A, equivalent to the investigation list L, a pointer j, and a boolean switch called "change." They initialize the optimizing pool of the entry node A[1] by the <u>0</u> element and proceed by the following steps :

<u>STEP 1</u> for j := 2 step 1 until k do

$$A[j] := \bigwedge_{d\ I(j)} f_d (A[d]);$$

where I(j) ={d | d is an immediate predecessor of J and d is less than j }

change := true;

while change do

Comment this while statement is equivalent to the step 2 of Kildall's algorithm:

if L is empty then halt;

begin

change := false;

<u>STEP 2</u> for j := 2 step 1 until k do

begin

$$temp := \bigwedge_{d\ I(j)} f_d (A[d]);$$

where I(j) = { d | d is an immediate predecessor of j }

```
if temp ≠ A[j] then change := true;
A[j] := temp;
end;
End of while;
```

Step two is equivalent to step four in Kildall's algorithm. A change of the optimizing pool of the currently processed node will cause a new iteration of the while statement which means a reprocessing of the node. This is accomplished by entering the node onto the investigation list again.

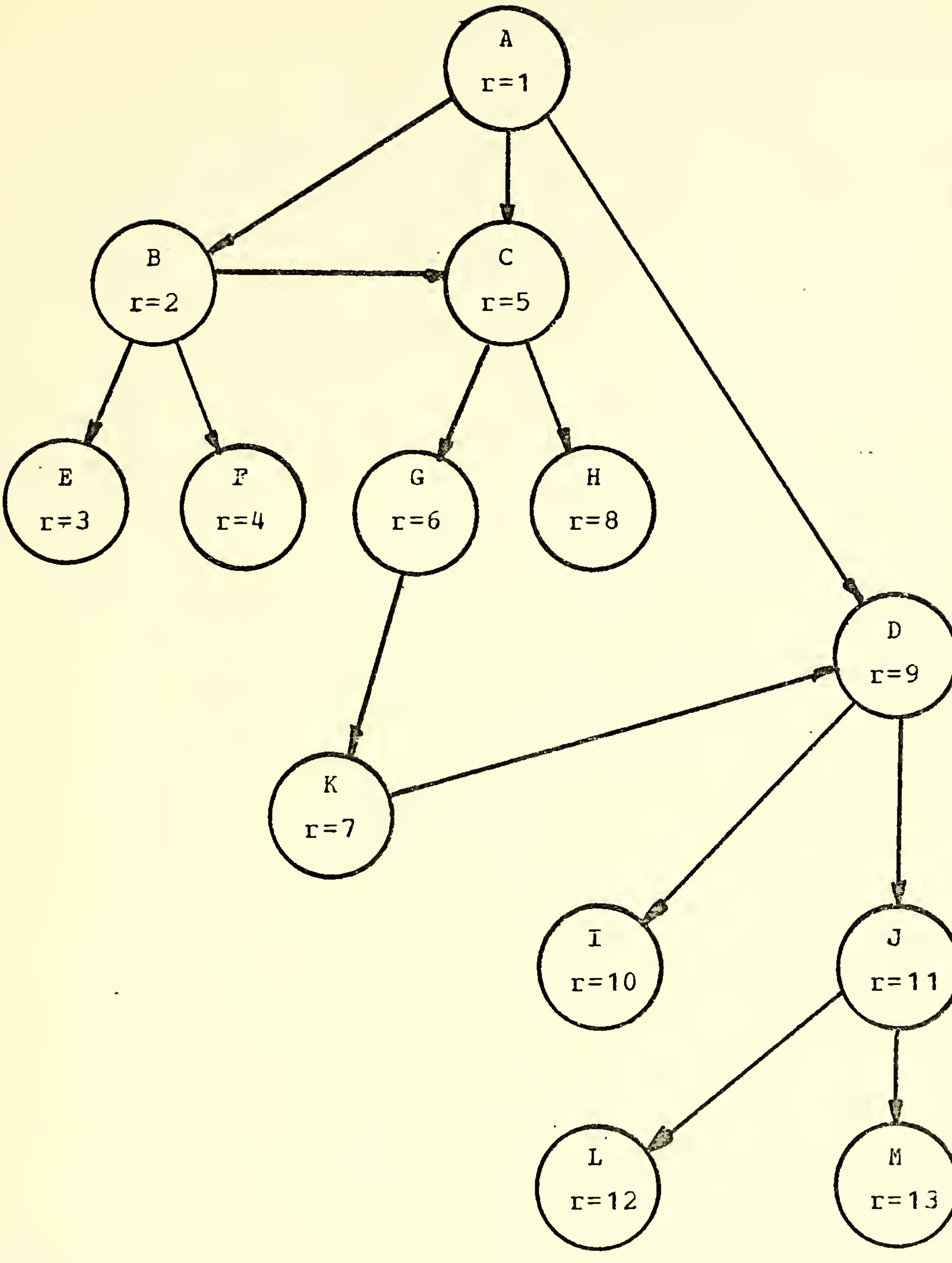


FIGURE 4.
Example for Ullman's Ordering Algorithm

## IV. IMPLEMENTATION OF THE SELECTION METHODS IN A SIMULATION

This section presents a simulation model for the purpose of evaluating the convergence rate of the various node selection methods. The model is based upon a randomly generated flow graph. The instructions attached to a node are represented by distances along the arc leading from a node to its successors. The optimizing pool in this model represents the sum of the arc lengths along the possible paths starting with the entry node up to the currently processed node. In this case the goal of the algorithm is to decrease the pool size in order to minimize the distance sum. This is done by searching along the possible paths to find the smallest sum.

The relation father - son and the associated arc length represented by the three tuple (NX,NY,NZ) is determined by a uniform random number generator[7]. To obtain the connectivity of the graph, the new father for the next level of successors is selected from the last group of sons.

The optimizing function maps the optimizing pool of the corresponding node into the output pool passed to the successors by adding the corresponding arc length NX to the optimizing pool:

Output pool = Optimizing pool + NX

In the general code optimization problem the optimizing function is applied only once when processing the node since the produced output pool is valid for all successors. In the model, however, the output pool differs due to the different arc lengths along the connections. To overcome this shortcoming of the model, all applications to compute the different output pools during processing the node are counted only as one application. This is accomplished by incrementing the "function tally" (a counter explained in section V) each time the corresponding node is processed.

The meet operator compares the incoming pool with the current optimizing pool and sets the new optimizing pool to the smaller of the two. If the old optimizing pool is substituted by a smaller one, the node will be entered into the investigation list if it does not already exist on the list. The list tally (counter explained in section V) is also incremented.

Due to the linear form of the optimizing function in the shortest path problem, the value of the optimizing pool is constantly increasing along a path. Therefore, each loop is only propagated once, except when additional paths lead to the loop entry, because the optimizing pool value at the end of the loop is always greater than or equal to the value at the beginning of the loop.

To make the simulation model closer to an actual program flow graph process, a second run will be made with the non-linear optimizing function

$$\text{Output pool} = \frac{OP^2}{NX^2} + \frac{OP}{NY} + NZ$$

where NX,NY,NZ are randomly generated coeffients attached to each arc. In this case, it is possible that the function value output is less than the incoming value OP. Therefore, the value at the end of a loop can be less than the value at the beginning and can cause additonal traversals of the loop.

The nonlinear optimizing function can be used, since the homomorphism condition is satisfied, which is shown in appendix A.

## V. EVALUATION CONCEPT

The final optimizing pool associated with each node upon termination of the algorithm is uniquely determined, independent of the order of choice, which is shown in Ref[6]. Thus, the number of steps leading to the final solution might be a measure of the convergence rate. Kildall[6] defined the upper bound on the number of steps for the given algorithm (section II-B) as follows:

Let n be the cardinality of N and h(OP) be a function of OP (which in turn may be a function of n) providing the maximum length of any chain between 1 and 0 in OP. Step 5 of Kildall's algorithm can be executed a maximum of h(OP) times for any given node. Since there are n nodes in the program graph, step 5 can be performed no more than
n * h(OP) times.

This upper bound on the number of steps is a theoretical one, and in actual practice the number of steps to a solution might be far less than that.

Ullman[3] proposed the number of iterations of the algorithm described in section III-D as the convergence rate measure. He claimed that the algorithm will halt after at most "d+3" iterations, where d is the maximum number of back edges given by any depth first spanning tree in a cycle free path.

Since the theoretical upper bound is only applicable to Kildall's algorithm, and the number of iterations is only applicable to Ullman's method these criteria are not uniformly meaningful for all four methods.

The evaluation concept used in the investigation described in the following section is based upon the application of the meet operator and the optimizing function first relative to the number of nodes, and then to the number of arcs. This is uniformly meaningful since the

implementation of the function and the operator is the same in all four methods. The methods will differ, however, in the number of the applications as already mentioned in their description. Since the convergence rate of the algorithm depends upon the number of applications of the meet and optimizing function, this number will be used as the evaluation criteria later on.

To determine the number of applications of the meet operation and optimizing function, two counters are associated with each node: a "list tally" denoted by LISTTALLY and a "function tally" denoted by FUNCTALLY.

The list tally is incremented each time the optimizing pool of the corresponding node is changed by the meet operator, i.e., the node is entered into the investigation list by the union operator.

The function tally will be incremented each time the optimizing pool of the corresponding node is mapped to an output pool using the optimizing function.

To compare the effect of the methods on the algorithm convergence rate, single node counters can be compared, and to get a general overview, the counters can be summed and the total number of applications during the algorithm run can be compared.

Using this evaluation concept the effects of the methods on the convergence rate can be investigated for different kind of graphs. The graphs investigated in the following paragraphs vary in:

the number of nodes,
the number of leaving arcs,
the number of generated levels,
and the arc length representing the different type of optimizing pools.

## VI. RESULTS

The effect of the node selection methods on the convergence rate of the algorithm when varying the simulation parameters is presented below.

The investigated graphs can be separated into five main groups, characterized by the number of nodes contained in the graph. The simulated graphs contained ten, twenty, thirty, fourty, and fifty nodes.

Each main group is divided into four subgroups distinguished by the number of generated levels contained in the corresponding graphs. The number of generated levels being considered are ten, twenty, thirty, and fifty.

Each subgroup contains five graphs which are equal in the number of nodes and the number of levels but vary in the numbers of leaving arcs. The number of leaving arcs being considered are two, four, ten, twenty, and thirty arcs.

The computer output starting on page 46 shows the results of the simulation runs using the linear and nonlinear optimizing functions. At the end of each main group the number of times a method used the lowest number of applications during the last twenty runs is presented.

Looking at the number of applications of the meet operator and optimizing function given by the corresponding tally will show the following trends.

Increasing the value of a variable generally caused an increase in the number of applications,

Increasing the number of nodes will cause on the average an increase on the number of applications in the following manner. Using the linear optimizing function the number of applications increases linearly when the steepest descent method is applied, and exponentially when the other methods are used. This is not surprising since one can easily show that each node is visited only once in the linear case when

using the steepest descent method. Using the non-linear optimizing function, the number of applications increases linearly in all four cases. The trends can be easily seen in Figure 5 for the linear function, and in Figure 6 for the non-linear function.

The increases caused by varying the number of levels and arcs are shown in Table 3 for graphs of ten and fifty nodes. The amount of increase when varying the number of arcs and levels depends also upon the number of nodes contained in the graph. This can be explained as follows: the size of a generated flow graph is limited by the number of nodes, levels, and arcs. The maximal number of distinct nodes being entered by distinct arcs is equal to the product of the number of levels and arcs. Whenever the value of this product is greater than the number of nodes of the corresponding graph, the generated new nodes being entered are already in the graph. That is, they are successors or predecessors of other nodes as well. After a certain increase in the number of levels or leaving arcs relative to the number of nodes, only loops or parallel arcs are generated. Increasing the number of loops or number of parallel arcs has little influence on the number of applications, since the probability that a successor's optimizing pool is already in its final state and the node is not reentered again into the investigation list increases also, which saves the application of the meet operator and optimizing function.

Looking at the results of the nonlinear function the same trend is noticed, but the number of applications is higher in general due to the additional traversals of the loops.

Looking at the results with regard to the effect of the different methods, the following can be observed.

The number of applications for the meet and optimizing function in the case of the LIFO method is always equal. This is neccessarily true since all nodes whose optimizing

pool was changed were entered into the list without using the union operator.

Investigating the list tally first to avoid the influence of the union operator will yield the following effects. With regard to graphs with small number of nodes and small numbers of leaving arcs the effect of the three methods LIFO, FIFO, and STEEP are similar, The number of applications of the meet operator varies only slightly. Increasing the number of levels or leaving arcs, however, the difference will be emphasized. The FIFO method used less applications than LIFO, but was outnumbered itself by the STEEP method which used the lowest number of applications in the most cases. The Ullman method is very handicapped by the "for statement" used inside the while loop. This influence became more obvious with increasing number of nodes in the investigated graphs.

The number of applications of the optimizing function depends very much on the number of meet operations, since only those nodes were considered which were on the investigation list. Therefore, when the number of meet operations decreases, the number of optimizing function applications also decreases. However, the improvement due to the union operator when entering a node into the investigation list is very obvious, since the differences between the list tally and function tally in all three methods FIFO, STEEP, and Ulmann are remarkable.

To get a general overview of the effect of the methods several runs of the same type were made with different seeds for the random number generator to provide a large number of distinct graphs. The results are shown in Table 4 using the linear function and in Table 5 using the nonlinear function. The tables contain the number of times the corresponding method used the lowest number of applications with regard to the meet and optimizing function. The summation of the corresponding numbers shows that the Steepest Descent method yields the best convergence rate.

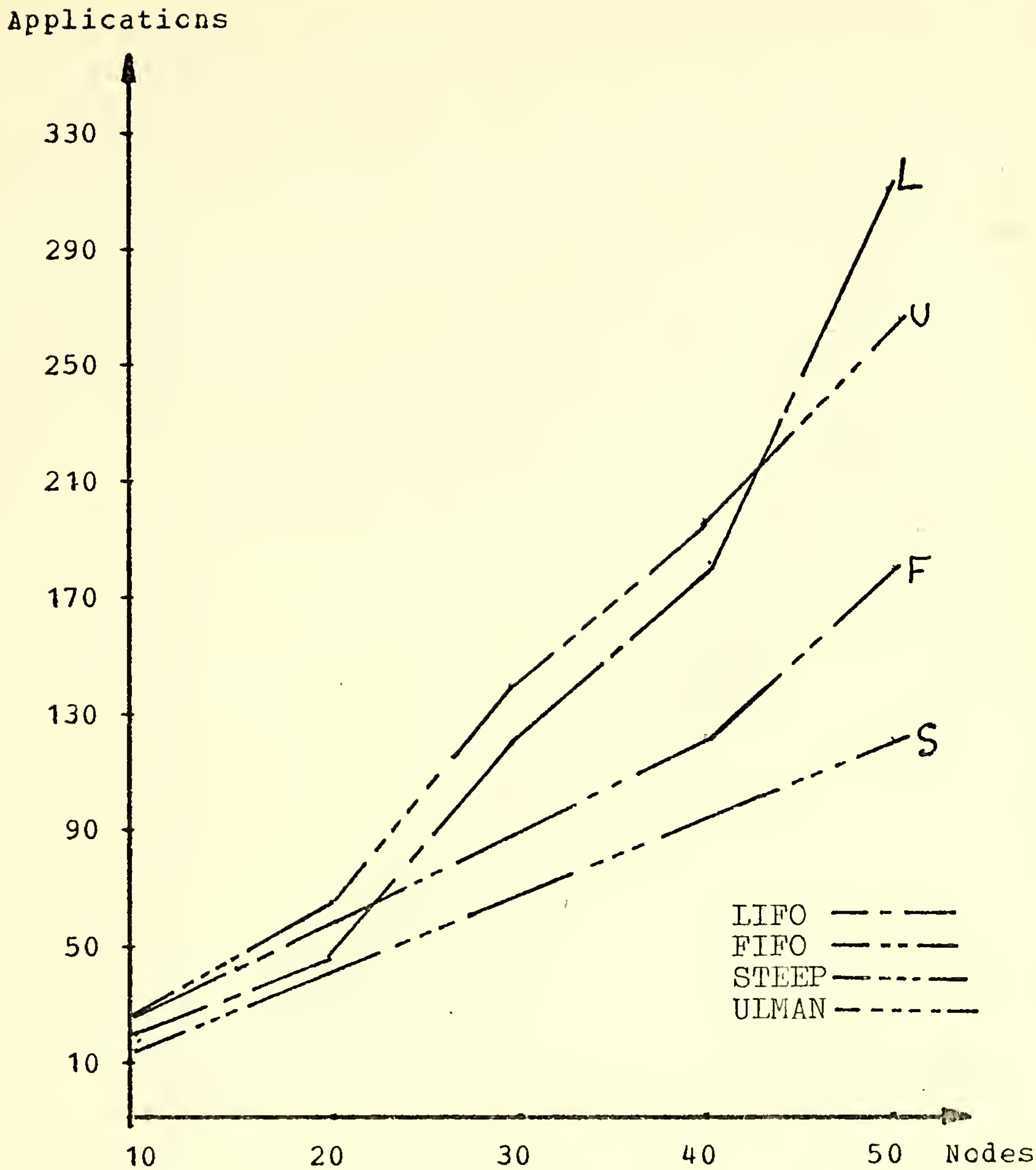


FIGURE 5.

Relation between the number of applications and number of nodes (linear function )

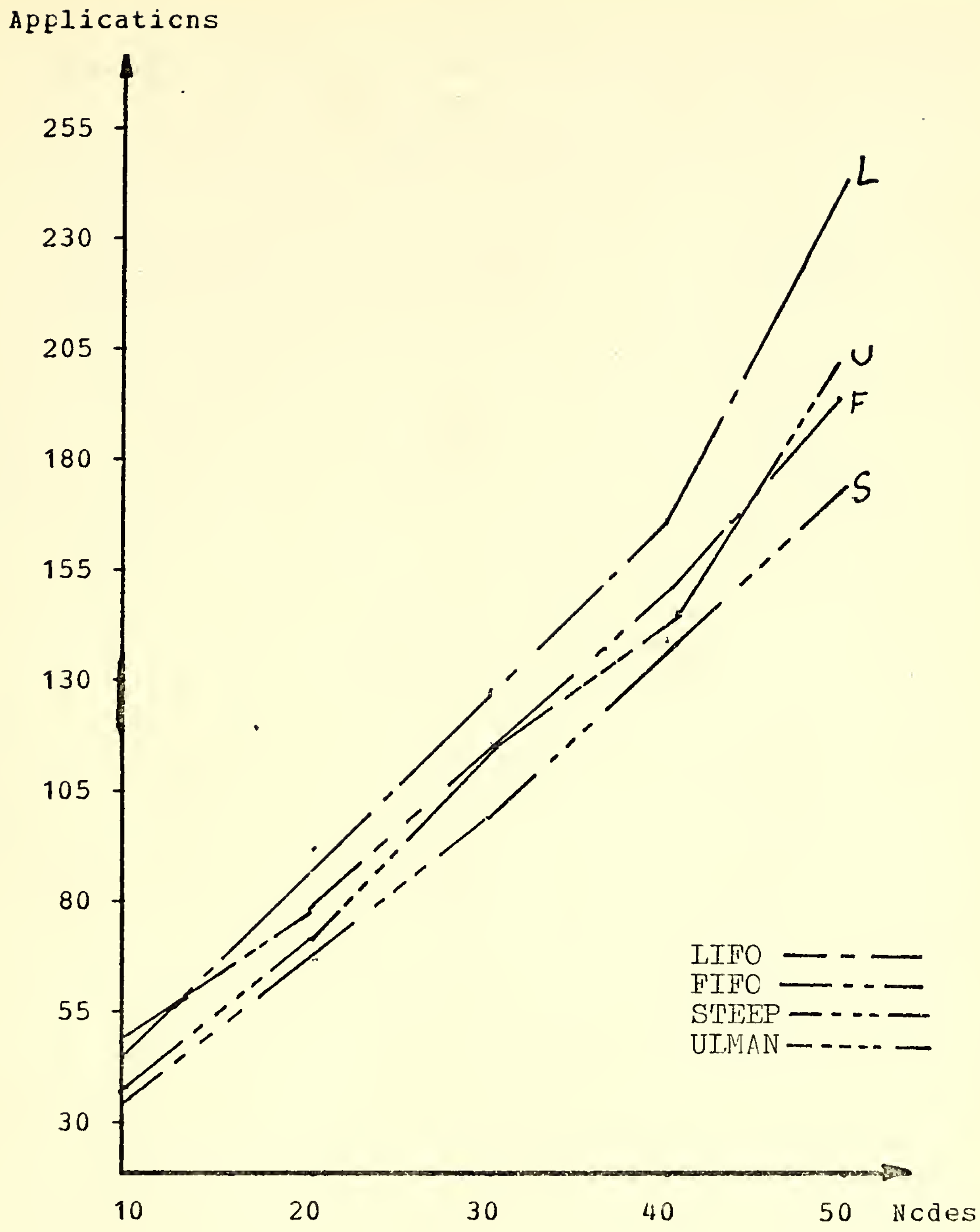


FIGURE 6.

Relation between the number of applications and number of nodes (non-linear function )

TABLE 3

Increases Caused by Changing the Simulation Parameters

Graph of ten nodes

| | level 10<br>arcs 2 | level 50<br>arcs 30 | increase |
|---|---|---|---|
| LIFO | 11 | 16 | 5 |
| FIFO | 10 | 18 | 8 |
| STEEP | 10 | 16 | 6 |
| ULLMAN | 17 | 19 | 2 |

Graph of fifty nodes

| | | | |
|---|---|---|---|
| LIFO | 16 | 304 | 288 |
| FIFO | 16 | 215 | 199 |
| STEEP | 16 | 131 | 115 |
| ULLMAN | 31 | 246 | 215 |

TABLE 4

Linear Function Results

| Seed | Orig | | Fifo | | Steep | | Ullman | |
|---|---|---|---|---|---|---|---|---|
| | List | Func | List | Func | List | Func | List | Func |
| | 10 | 0 | 5 | 0 | 17 | 18 | 1 | 2 |
| | 2 | 1 | 1 | 1 | 19 | 20 | 0 | 0 |
| 12345678 | 1 | 1 | 5 | 4 | 19 | 20 | 0 | 0 |
| | 1 | 0 | 3 | 2 | 20 | 19 | 0 | 1 |
| | 1 | 1 | 1 | 1 | 20 | 20 | 0 | 1 |
| | 1 | 1 | 4 | 1 | 19 | 20 | 0 | 1 |
| 87654321 | 1 | 1 | 2 | 3 | 20 | 20 | 0 | 1 |
| | 1 | 0 | 3 | 3 | 19 | 19 | 0 | 1 |
| | 1 | 0 | 2 | 1 | 20 | 19 | 0 | 1 |
| | 6 | 0 | 3 | 1 | 20 | 20 | 1 | 1 |
| | 1 | 0 | 1 | 1 | 18 | 20 | 0 | 0 |
| 777882234 | 0 | 0 | 2 | 3 | 20 | 19 | 0 | 1 |
| | 1 | 1 | 1 | 2 | 20 | 20 | 0 | 0 |
| | 2 | 2 | 2 | 4 | 19 | 20 | 0 | 1 |
| | 10 | 1 | 4 | 2 | 18 | 20 | 0 | 1 |
| | 0 | 0 | 3 | 3 | 19 | 20 | 0 | 0 |
| 15376482 | 3 | 0 | 3 | 2 | 19 | 19 | 0 | 1 |
| | 1 | 1 | 2 | 2 | 20 | 20 | 0 | 2 |
| | 2 | 0 | 3 | 0 | 19 | 19 | 0 | 1 |
| Summation : | | | | | | | | |
| | 55 | 11 | 54 | 38 | 380 | 392 | 2 | 18 |

TABLE 5

Nonlinear Function Results

| Seed | Orig List | Orig Func | Fifo List | Fifo Func | Steep List | Steep Func | Ullman List | Ullman Func |
|---|---|---|---|---|---|---|---|---|
| | 3 | 0 | 3 | 4 | 12 | 18 | 5 | 0 |
| | 1 | 1 | 2 | 2 | 18 | 19 | 2 | 1 |
| 12345678 | 6 | 0 | 4 | 2 | 11 | 15 | 0 | 3 |
| | 4 | 1 | 4 | 2 | 15 | 18 | 0 | 3 |
| | 5 | 0 | 5 | 3 | 15 | 18 | 2 | 0 |
| | 2 | 0 | 2 | 2 | 15 | 19 | 3 | 0 |
| 87654321 | 4 | 1 | 2 | 0 | 15 | 19 | 0 | 2 |
| | 3 | 0 | 3 | 2 | 16 | 18 | 1 | 1 |
| | 4 | 0 | 2 | 0 | 17 | 18 | 0 | 2 |
| | 4 | 1 | 5 | 11 | 8 | 11 | 5 | 1 |
| | 2 | 0 | 4 | 2 | 13 | 17 | 2 | 1 |
| 777882234 | 4 | 0 | 4 | 1 | 15 | 18 | 0 | 1 |
| | 3 | 1 | 2 | 0 | 14 | 18 | 1 | 2 |
| | 7 | 2 | 1 | 3 | 12 | 16 | 1 | 3 |
| | 4 | 0 | 5 | 7 | 9 | 16 | 4 | 0 |
| | 1 | 0 | 1 | 2 | 16 | 19 | 4 | 0 |
| 15376482 | 3 | 0 | 3 | 1 | 18 | 20 | 0 | 0 |
| | 4 | 1 | 6 | 2 | 13 | 17 | 0 | 3 |
| | 2 | 1 | 3 | 2 | 17 | 18 | 0 | 3 |
| Summation : | | | | | | | | |
| | 69 | 11 | 63 | 50 | 286 | 350 | 30 | 27 |

# VII. CONCLUSIONS

The effect of the four selection methods: Last In First Out, First In First Out, Steepest Descent, and Ullman's Depth First Search was investigated with regard to the convergence rate of Kildall's program flow graph analysis algorithm. The overall result was that the steepest descent method required the smallest number of applications. One question which arises is how to implement the idea of the steepest descent for a real program? The amount of computations to determine the "smallest optimizing pool" might be more expensive with regard to execution time or storage to process the algorithm then the effort to optimize by the other methods. This possible disadvantage can be overcome by the following. When computing the meet operation the size value of the pool can be computed too. This size value has to be stored for later use and will occupy additional memory space. Therefore, if a small number of applications of the meet or optimizing functions is required, the steepest descent method is most efficient, if an easy implementation is required, the first in first out method will be more easy to apply.

The graphs investigated in these simulation runs varied in the number of nodes, the number of levels, and the number of leaving arcs. The results were uniform and nearly independent of the graph type. Therefore the result of a real program investigation should not differ substantially from the results of this simulation.

## APPENDIX A

The non - linear optimizing function is applicable, since the homomorphism condition is satisfied. That is, given that

$$f(n,P) = \frac{P^2}{NX^2} + \frac{P}{NY} + NZ, \quad \text{and}$$

$$P_1 \wedge P_2 = \text{Min}\,(P_1, P_2)$$

then
$$f(n, P_1 \wedge P_2) = f(n,P_1) \wedge f(n,P_2).$$

This is shown to be correct in the following way:

$$f(n,P_1 \wedge P_2) = \frac{(P_1 \wedge P_2)^2}{NX^2} + \frac{(P_1 \wedge P_2)}{NY} + NZ,$$

$$f(n,P_1) = \frac{P_1^{\;2}}{NX^2} + \frac{P_1}{NY} + NZ, \quad \text{and}$$

$$f(n,P_2) = \frac{P_2^{\;2}}{NX^2} + \frac{P_2}{NY} + NZ.$$

Thus

$$f((n,P_1) \wedge (n,P_2)) = \text{Min}\left(\frac{P_1^{\;2}}{NX^2} + \frac{P_1}{NY} + NZ, \right.$$

$$\left. \frac{P_2^{\;2}}{NX^2} + \frac{P_2}{NY} + NZ\right)$$

$$= \text{Min}\left(\frac{P_1^{\;2}}{NX^2} + \frac{P_1}{NY}\ ,\ \frac{P_2^{\;2}}{NX^2} + \frac{P_2}{NY}\right) + NZ.$$

Since $P_1, P_2 \geq 0$

$$= \text{Min}\left(\frac{P_1^{\;2}}{NX^2}\ ,\ \frac{P_2^{\;2}}{NX^2}\right) + \text{Min}\left(\frac{P_1}{NY}\ ,\ \frac{P_2}{NY}\right) + NZ,$$

Since NX and NY are constants

$$= \frac{\mathrm{Min}(P_1{}^2, P_2{}^2)}{NX^2} + \frac{\mathrm{Min}(P_1, P_2)}{NY} + NZ$$

$$= \frac{\mathrm{Min}(P_1, P_2)^2}{NX^2} + \frac{\mathrm{Min}(P_1, P_2)}{NY} + NZ$$

$$= \frac{(P_1 \wedge P_2)^2}{NX^2} + \frac{(P_1 \wedge P_2)}{NY} + NZ$$

$= f(n, P_1 \wedge P_2)$ which is the required identity.

COMPUTER OUTPUT

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE LINEAR OPTIMIZING FUNCTION WAS USED

| RUN | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NMNODE | | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 10 | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 20 |

THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 11 | 12 | 14 | 13 | 18 | 14 | 17 | 20 | 20 | 17 |
| | FUNCTALLY | 11 | 12 | 14 | 13 | 18 | 14 | 17 | 20 | 20 | 17 |
| FIFO | LISTTALLY | 10 | 11 | 18 | 19 | 19 | 15 | 14 | 23 | 28 | 22 |
| | FUNCTALLY | 9 | 10 | 13 | 13 | 14 | 11 | 12 | 14 | 20 | 15 |
| STEEP | LISTTALLY | 10 | 10 | 16 | 13 | 17 | 15 | 14 | 20 | 21 | 16 |
| | FUNCTALLY | 9 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| ULLMAN | LISTTALLY | 17 | 19 | 19 | 19 | 28 | 28 | 28 | 28 | 19 | 19 |
| | FUNCTALLY | 8 | 9 | 17 | 17 | 20 | 13 | 15 | 19 | 18 | 17 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE LINEAR OPTIMIZING FUNCTION WAS USED

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RUN | | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| NMNODE | | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 30 | 30 | 30 | 30 | 30 | 50 | 50 | 50 | 50 | 50 |

THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 16 | 18 | 26 | 20 | 22 | 23 | 22 | 18 | 14 | 16 |
| | FUNCTALLY | 16 | 18 | 26 | 20 | 22 | 23 | 22 | 18 | 14 | 16 |
| FIFO | LISTTALLY | 16 | 18 | 26 | 22 | 23 | 20 | 26 | 14 | 14 | 18 |
| | FUNCTALLY | 12 | 13 | 18 | 14 | 12 | 13 | 17 | 12 | 13 | 16 |
| STEEP | LISTTALLY | 13 | 16 | 26 | 20 | 22 | 19 | 22 | 14 | 14 | 16 |
| | FUNCTALLY | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| ULLMAN | LISTTALLY | 37 | 28 | 28 | 28 | 28 | 37 | 37 | 28 | 19 | 19 |
| | FUNCTALLY | 17 | 18 | 24 | 21 | 25 | 23 | 26 | 14 | 13 | 17 |

THE FOLLOWING LIST REPRESENTS THE NUMBER
OF TIMES THE CORRESPONDING METHOD USED THE
LOWEST OR SAME NUMBER OF APPLICATIONS
DURING THE LAST 20 RUNS.

| | ORIG | FIFO | STEEP | ULLMAN |
|---|---|---|---|---|
| LISTTALLY | 10 | 5 | 17 | 1 |
| FUNCTALLY | 0 | 0 | 18 | 2 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE LINEAR CPTIMIZING FUNCTION WAS USED

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RUN | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| NMNODE | | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 10 | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 20 |

THE FOLLCWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 13 | 23 | 30 | 66 | 36 | 22 | 34 | 42 | 49 | 49 |
| | FUNCTALLY | 13 | 23 | 30 | 66 | 36 | 22 | 34 | 42 | 49 | 49 |
| FIFO | LISTTALLY | 13 | 23 | 33 | 47 | 38 | 23 | 33 | 40 | 55 | 57 |
| | FUNCTALLY | 13 | 20 | 24 | 29 | 27 | 20 | 29 | 23 | 28 | 38 |
| STEEP | LISTTALLY | 13 | 21 | 26 | 42 | 34 | 17 | 23 | 30 | 35 | 45 |
| | FUNCTALLY | 13 | 17 | 20 | 20 | 20 | 16 | 19 | 20 | 20 | 20 |
| ULLMAN | LISTTALLY | 25 | 65 | 39 | 96 | 39 | 46 | 73 | 77 | 58 | 77 |
| | FUNCTALLY | 14 | 25 | 33 | 50 | 37 | 23 | 29 | 36 | 55 | 56 |

```
STATISTICAL SUMMARY
STARTSEED FOR THIS  RUN =    12345678
THE LINEAR OPTIMIZING FUNCTION WAS USED
RUN                    11  12  13  14  15  16  17  18  19  20
NMNODE                 20  20  20  20  20  20  20  20  20  20
MAXCON                  2   4  10  20  30   2   4  10  20  30
MALGTH                100 100 100 100 100 100 100 100 100 100
MAXLVL                 30  30  30  30  30  50  50  50  50  50
THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.
LIFO    LISTTALLY      23  45  54  57  77  49  53  58  71  55
        FUNCTALLY      23  45  54  57  77  49  53  58  71  55
FIFO    LISTTALLY      25  40  37  52  78  38  53  56  82  77
        FUNCTALLY      23  26  26  29  40  26  34  33  45  42
STEEP   LISTTALLY      21  28  30  42  49  29  40  50  50  57
        FUNCTALLY      17  20  20  20  20  20  20  20  20  20
ULLMAN  LISTTALLY      49  77  96  96  77  58  96  77  58  58
        FUNCTALLY      23  46  43  53  70  39  53  51  55  71


THE FOLLOWING LIST REPRESENTS THE NUMBER
OF TIMES THE CORRESPONDING METHOD USED THE
LOWEST  OR SAME NUMBER OF APPLICATIONS
DURING THE LAST 20 RUNS.
                      ORIG    FIFO   STEEP  ULLMAN
LISTTALLY                2       1      19       0
FUNCTALLY                1       1      20       0
```

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE LINEAR OPTIMIZING FUNCTION WAS USED

| RUN | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NMNODE | | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 10 | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 20 |

THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 17 | 27 | 46 | 60 | 65 | 31 | 63 | 89 | 105 | 86 |
| | FUNCTALLY | 17 | 27 | 46 | 60 | 65 | 31 | 63 | 89 | 105 | 86 |
| FIFO | LISTTALLY | 17 | 25 | 40 | 41 | 63 | 28 | 33 | 53 | 66 | 74 |
| | FUNCTALLY | 17 | 24 | 30 | 34 | 44 | 27 | 30 | 32 | 38 | 49 |
| STEEP | LISTTALLY | 17 | 25 | 35 | 41 | 54 | 27 | 36 | 44 | 54 | 54 |
| | FUNCTALLY | 17 | 24 | 29 | 30 | 30 | 27 | 29 | 30 | 30 | 30 |
| ULLMAN | LISTTALLY | 33 | 47 | 85 | 88 | 88 | 53 | 85 | 117 | 146 | 117 |
| | FUNCTALLY | 18 | 28 | 40 | 53 | 57 | 28 | 40 | 66 | 80 | 66 |

STATISTICAL SUMMARY

STARTSEED FCR THIS RUN = 12345678

THE LINEAR OPTIMIZING FUNCTION WAS USED

| RUN | | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NMNODE | | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 30 | 30 | 30 | 30 | 30 | 50 | 50 | 50 | 50 | 50 |

THE FOLLCWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 59 | 85 | 136 | 117 | 89 | 98 | 106 | 87 | 140 | 139 |
| | FUNCTALLY | 59 | 85 | 136 | 117 | 89 | 98 | 106 | 87 | 140 | 139 |
| FIFO | LISTTALLY | 28 | 51 | 75 | 93 | 107 | 55 | 71 | 95 | 91 | 95 |
| | FUNCTALLY | 24 | 36 | 41 | 52 | 55 | 47 | 51 | 48 | 41 | 48 |
| STEEP | LISTTALLY | 28 | 37 | 52 | 56 | 75 | 39 | 47 | 61 | 80 | 78 |
| | FUNCTALLY | 24 | 29 | 30 | 30 | 30 | 29 | 30 | 30 | 30 | 30 |
| ULLMAN | LISTTALLY | 93 | 113 | 117 | 175 | 88 | 113 | 117 | 117 | 146 | 175 |
| | FUNCTALLY | 36 | 60 | 92 | 72 | 84 | 64 | 76 | 71 | 101 | 121 |

THE FOLLCWING LIST REPRESENTS THE NUMBER
OF TIMES THE CORRESPONDING METHOD USED THE
LCWEST OR SAME NUMBER OF APPLICATIONS
DURING THE LAST 20 RUNS.

| | ORIG | FIFO | STEEP | ULLMAN |
|---|---|---|---|---|
| LISTTALLY | 1 | 5 | 19 | 0 |
| FUNCTALLY | 1 | 4 | 20 | 0 |

STATISTICAL SUMMARY

STARTSEED FCR THIS RUN = 12345678

THE LINEAR OPTIMIZING FUNCTION WAS USED

| RUN | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NMNODE | | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 |
| MAXCCN | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 10 | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 20 |

THE FOLLCWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 23 | 27 | 94 | 87 | 72 | 39 | 73 | 112 | 126 | 141 |
| | FUNCTALLY | 23 | 27 | 94 | 87 | 72 | 39 | 73 | 112 | 126 | 141 |
| FIFO | LISTTALLY | 16 | 27 | 52 | 66 | 62 | 27 | 46 | 81 | 86 | 118 |
| | FUNCTALLY | 16 | 26 | 43 | 56 | 44 | 24 | 39 | 56 | 49 | 57 |
| STEEP | LISTTALLY | 16 | 27 | 47 | 50 | 62 | 26 | 37 | 56 | 67 | 89 |
| | FUNCTALLY | 16 | 26 | 36 | 40 | 40 | 24 | 33 | 40 | 40 | 40 |
| ULLMAN | LISTTALLY | 31 | 51 | 106 | 157 | 118 | 70 | 97 | 118 | 157 | 196 |
| | FUNCTALLY | 15 | 27 | 48 | 77 | 70 | 28 | 48 | 82 | 113 | 127 |

STATISTICAL SUMMARY

STARTSEED FCR THIS RUN = 12345678

THE LINEAR OPTIMIZING FUNCTION WAS USED

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RUN | | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| NMNODE | | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 30 | 30 | 30 | 30 | 30 | 50 | 50 | 50 | 50 | 50 |

THE FOLLCWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 62 | 88 | 123 | 131 | 142 | 206 | 107 | 158 | 149 | 179 |
| | FUNCTALLY | 62 | 88 | 123 | 131 | 142 | 206 | 107 | 158 | 149 | 179 |
| FIFC | LISTTALLY | 42 | 49 | 91 | 101 | 128 | 54 | 64 | 82 | 124 | 142 |
| | FUNCTALLY | 40 | 42 | 66 | 57 | 63 | 44 | 48 | 52 | 63 | 69 |
| STEEP | LISTTALLY | 33 | 44 | 61 | 75 | 89 | 45 | 52 | 72 | 94 | 97 |
| | FUNCTALLY | 31 | 39 | 40 | 40 | 40 | 38 | 38 | 40 | 40 | 40 |
| ULLMAN | LISTTALLY | 91 | 153 | 235 | 157 | 196 | 149 | 149 | 196 | 157 | 196 |
| | FUNCTALLY | 48 | 67 | 94 | 100 | 131 | 76 | 86 | 98 | 120 | 126 |

THE FCLLCWING LIST REPRESENTS THE NUMBER
OF TIMES THE CORRESPONDING METHOD USED THE
LOWEST OR SAME NUMBER OF APPLICATIONS
DURING THE LAST 20 RUNS.

| | ORIG | FIFO | STEEP | ULLMAN |
|---|---|---|---|---|
| LISTTALLY | 1 | 3 | 20 | 0 |
| FUNCTALLY | 0 | 2 | 19 | 1 |

STATISTICAL SUMMARY

STARTSEED FCR THIS RUN = 12345678

THE LINEAR OPTIMIZING FUNCTION WAS USED

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RUN | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| NMNODE | | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 10 | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 20 |

THE FOLLCWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 16 | 36 | 53 | 66 | 161 | 34 | 92 | 88 | 106 | 112 |
| | FUNCTALLY | 16 | 36 | 53 | 66 | 161 | 34 | 92 | 88 | 106 | 112 |
| FIFO | LISTTALLY | 16 | 32 | 54 | 88 | 77 | 34 | 44 | 80 | 120 | 118 |
| | FUNCTALLY | 16 | 31 | 51 | 70 | 54 | 34 | 42 | 61 | 77 | 76 |
| STEEP | LISTTALLY | 16 | 31 | 46 | 62 | 65 | 31 | 43 | 65 | 84 | 97 |
| | FUNCTALLY | 16 | 29 | 43 | 49 | 50 | 30 | 39 | 50 | 50 | 50 |
| ULLMAN | LISTTALLY | 31 | 57 | 127 | 145 | 148 | 59 | 115 | 197 | 197 | 148 |
| | FUNCTALLY | 16 | 31 | 57 | 80 | 79 | 31 | 66 | 93 | 114 | 102 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE LINEAR OPTIMIZING FUNCTION WAS USED

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RUN | | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| NMNODE | | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 |
| MAXCON | | 2 | 4 | 10 | 20 | 2 | 4 | 10 | 20 | 30 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 30 | 30 | 30 | 30 | 50 | 50 | 50 | 50 | 30 | 50 |

THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 46 | 87 | 148 | 214 | 169 | 190 | 214 | 264 | 232 | 304 |
| | FUNCTALLY | 46 | 87 | 148 | 214 | 169 | 190 | 214 | 264 | 232 | 304 |
| FIFO | LISTTALLY | 51 | 75 | 96 | 103 | 54 | 90 | 109 | 157 | 149 | 215 |
| | FUNCTALLY | 48 | 65 | 61 | 64 | 49 | 66 | 76 | 84 | 93 | 108 |
| STEEP | LISTTALLY | 35 | 53 | 69 | 98 | 49 | 67 | 86 | 121 | 118 | 131 |
| | FUNCTALLY | 31 | 46 | 50 | 50 | 43 | 50 | 50 | 50 | 50 | 50 |
| ULLMAN | LISTTALLY | 121 | 136 | 246 | 197 | 127 | 197 | 246 | 246 | 246 | 246 |
| | FUNCTALLY | 58 | 78 | 116 | 112 | 61 | 91 | 119 | 176 | 165 | 186 |

THE FOLLOWING LIST REPRESENTS THE NUMBER
OF TIMES THE CORRESPONDING METHOD USED THE
LOWEST OR SAME NUMBER OF APPLICATIONS
DURING THE LAST 20 RUNS.

| | ORIG | FIFO | STEEP | ULLMAN |
|---|---|---|---|---|
| LISTTALLY | 1 | 1 | 20 | 0 |
| FUNCTALLY | 1 | 1 | 20 | 1 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE NONLINEAR OPTIMIZING FUNCTION WAS USED

| RUN | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NMNODE | | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 10 | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 20 |

THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 11 | 24 | 27 | 26 | 42 | 19 | 27 | 39 | 40 | 39 |
| | FUNCTALLY | 11 | 24 | 27 | 26 | 42 | 19 | 27 | 39 | 40 | 39 |
| FIFO | LISTTALLY | 13 | 20 | 18 | 26 | 46 | 17 | 22 | 31 | 44 | 36 |
| | FUNCTALLY | 12 | 15 | 16 | 15 | 22 | 14 | 15 | 20 | 21 | 22 |
| STEEP | LISTTALLY | 14 | 20 | 20 | 22 | 39 | 15 | 30 | 31 | 40 | 33 |
| | FUNCTALLY | 12 | 14 | 15 | 10 | 16 | 11 | 18 | 17 | 15 | 18 |
| ULLMAN | LISTTALLY | 25 | 37 | 46 | 37 | 37 | 28 | 37 | 37 | 37 | 28 |
| | FUNCTALLY | 11 | 23 | 19 | 26 | 34 | 15 | 23 | 32 | 44 | 35 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE NONLINEAR OPTIMIZING FUNCTION WAS USED

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RUN | | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| NMNODE | | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 30 | 30 | 30 | 30 | 30 | 50 | 50 | 50 | 50 | 50 |

THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 27 | 29 | 35 | 46 | 38 | 32 | 40 | 35 | 38 | 48 |
| | FUNCTALLY | 27 | 29 | 35 | 46 | 38 | 32 | 40 | 35 | 38 | 48 |
| FIFO | LISTTALLY | 24 | 30 | 39 | 37 | 32 | 26 | 37 | 42 | 35 | 51 |
| | FUNCTALLY | 18 | 16 | 18 | 19 | 15 | 17 | 18 | 22 | 18 | 16 |
| STEEP | LISTTALLY | 20 | 25 | 24 | 32 | 27 | 21 | 27 | 38 | 42 | 39 |
| | FUNCTALLY | 15 | 17 | 10 | 16 | 17 | 10 | 14 | 17 | 25 | 20 |
| ULLMAN | LISTTALLY | 37 | 37 | 37 | 37 | 37 | 28 | 37 | 37 | 46 | 37 |
| | FUNCTALLY | 22 | 28 | 31 | 37 | 29 | 22 | 33 | 38 | 33 | 48 |

THE FOLLOWING LIST REPRESENTS THE NUMBER
OF TIMES THE CORRESPONDING METHOD USED THE
LOWEST OR SAME NUMBER OF APPLICATIONS
DURING THE LAST 20 RUNS.

| | ORIG | FIFO | STEEP | ULLMAN |
|---|---|---|---|---|
| LISTTALLY | 2 | 5 | 11 | 4 |
| FUNCTALLY | 1 | 5 | 14 | 1 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE NONLINEAR OPTIMIZING FUNCTION WAS USED

| RUN | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NMNODE | | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 10 | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 20 |

THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 27 | 27 | 39 | 37 | 66 | 31 | 36 | 72 | 101 | 89 |
| | FUNCTALLY | 27 | 27 | 39 | 37 | 66 | 31 | 36 | 72 | 101 | 89 |
| FIFO | LISTTALLY | 17 | 23 | 46 | 48 | 61 | 23 | 37 | 47 | 86 | 70 |
| | FUNCTALLY | 16 | 20 | 31 | 35 | 36 | 20 | 29 | 30 | 43 | 37 |
| STEEP | LISTTALLY | 17 | 21 | 43 | 41 | 52 | 29 | 31 | 43 | 61 | 73 |
| | FUNCTALLY | 16 | 16 | 26 | 21 | 28 | 23 | 24 | 21 | 24 | 32 |
| ULLMAN | LISTTALLY | 40 | 46 | 39 | 58 | 58 | 52 | 58 | 77 | 77 | 77 |
| | FUNCTALLY | 15 | 29 | 37 | 41 | 60 | 30 | 32 | 50 | 73 | 73 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE NONLINEAR OPTIMIZING FUNCTION WAS USED

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RUN | | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| NMNODE | | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| MAXCCN | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 30 | 30 | 30 | 30 | 30 | 50 | 50 | 50 | 50 | 50 |

THE FOLLCWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 49 | 59 | 88 | 74 | 119 | 60 | 64 | 72 | 119 | 110 |
| | FUNCTALLY | 49 | 59 | 88 | 74 | 119 | 60 | 64 | 72 | 119 | 110 |
| FIFO | LISTTALLY | 54 | 55 | 57 | 61 | 90 | 47 | 64 | 84 | 72 | 106 |
| | FUNCTALLY | 46 | 34 | 34 | 42 | 46 | 33 | 42 | 40 | 34 | 38 |
| STEEP | LISTTALLY | 43 | 58 | 56 | 68 | 84 | 42 | 60 | 61 | 77 | 82 |
| | FUNCTALLY | 27 | 34 | 34 | 38 | 28 | 25 | 37 | 24 | 27 | 30 |
| ULLMAN | LISTTALLY | 73 | 77 | 96 | 96 | 77 | 77 | 96 | 58 | 77 | 77 |
| | FUNCTALLY | 43 | 57 | 67 | 62 | 76 | 44 | 64 | 64 | 91 | 94 |

THE FOLLCWING LIST REPRESENTS THE NUMBER
OF TIMES THE CORRESPONDING METHOD USED THE
LOWEST OR SAME NUMBER OF APPLICATIONS
DURING THE LAST 20 RUNS.

| | ORIG | FIFO | STEEP | ULLMAN |
|---|---|---|---|---|
| LISTTALLY | 2 | 6 | 10 | 4 |
| FUNCTALLY | 0 | 3 | 18 | 1 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE NONLINEAR OPTIMIZING FUNCTION WAS USED

| RUN | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NMNODE | | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 10 | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 20 |

THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 19 | 49 | 51 | 81 | 87 | 33 | 52 | 66 | 132 | 137 |
| | FUNCTALLY | 19 | 49 | 51 | 81 | 87 | 33 | 52 | 66 | 132 | 137 |
| FIFO | LISTTALLY | 19 | 37 | 67 | 83 | 82 | 31 | 49 | 65 | 99 | 108 |
| | FUNCTALLY | 19 | 35 | 52 | 52 | 56 | 29 | 42 | 47 | 53 | 63 |
| STEEP | LISTTALLY | 19 | 33 | 52 | 71 | 67 | 34 | 53 | 68 | 87 | 104 |
| | FUNCTALLY | 19 | 27 | 34 | 43 | 34 | 29 | 38 | 42 | 45 | 42 |
| ULLMAN | LISTTALLY | 37 | 70 | 85 | 88 | 88 | 58 | 105 | 88 | 146 | 117 |
| | FUNCTALLY | 19 | 36 | 54 | 85 | 87 | 30 | 48 | 67 | 97 | 109 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE NONLINEAR OPTIMIZING FUNCTION WAS USED

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RUN | | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| NMNODE | | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 30 | 30 | 30 | 30 | 30 | 50 | 50 | 50 | 50 | 50 |

THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 54 | 90 | 101 | 132 | 116 | 78 | 82 | 120 | 144 | 155 |
| | FUNCTALLY | 54 | 90 | 101 | 132 | 116 | 78 | 82 | 120 | 144 | 155 |
| FIFO | LISTTALLY | 47 | 61 | 82 | 122 | 109 | 60 | 82 | 99 | 149 | 151 |
| | FUNCTALLY | 38 | 43 | 44 | 67 | 63 | 49 | 52 | 52 | 62 | 57 |
| STEEP | LISTTALLY | 53 | 64 | 65 | 106 | 95 | 61 | 73 | 96 | 99 | 122 |
| | FUNCTALLY | 43 | 38 | 37 | 43 | 42 | 38 | 41 | 44 | 32 | 43 |
| ULLMAN | LISTTALLY | 70 | 117 | 88 | 117 | 88 | 117 | 117 | 117 | 117 | 117 |
| | FUNCTALLY | 37 | 70 | 79 | 115 | 94 | 60 | 85 | 103 | 107 | 143 |

THE FOLLOWING LIST REPRESENTS THE NUMBER
OF TIMES THE CORRESPONDING METHOD USED THE
LOWEST OR SAME NUMBER OF APPLICATIONS
DURING THE LAST 20 RUNS.

| | ORIG | FIFO | STEEP | ULLMAN |
|---|---|---|---|---|
| LISTTALLY | 2 | 7 | 11 | 2 |
| FUNCTALLY | 1 | 2 | 19 | 2 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE NONLINEAR OPTIMIZING FUNCTION WAS USED

| RUN | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NMNODE | | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 10 | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 20 |

THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 21 | 38 | 82 | 84 | 125 | 45 | 58 | 88 | 144 | 176 |
| | FUNCTALLY | 21 | 38 | 82 | 84 | 125 | 45 | 58 | 88 | 144 | 176 |
| FIFO | LISTTALLY | 21 | 40 | 84 | 86 | 115 | 49 | 63 | 86 | 140 | 137 |
| | FUNCTALLY | 21 | 38 | 68 | 71 | 72 | 48 | 56 | 61 | 80 | 78 |
| STEEP | LISTTALLY | 21 | 39 | 73 | 89 | 96 | 42 | 62 | 85 | 113 | 121 |
| | FUNCTALLY | 20 | 37 | 53 | 67 | 49 | 36 | 48 | 58 | 60 | 58 |
| ULLMAN | LISTTALLY | 49 | 82 | 100 | 157 | 118 | 88 | 109 | 157 | 157 | 157 |
| | FUNCTALLY | 19 | 35 | 71 | 90 | 111 | 48 | 54 | 90 | 121 | 124 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE NONLINEAR OPTIMIZING FUNCTION WAS USED

| RUN | | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NMNODE | | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 30 | 30 | 30 | 30 | 30 | 50 | 50 | 50 | 50 | 50 |

THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LIFO | LISTTALLY | 57 | 74 | 104 | 135 | 193 | 58 | 92 | 139 | 172 | 185 |
| | FUNCTALLY | 57 | 74 | 104 | 135 | 193 | 58 | 92 | 139 | 172 | 185 |
| FIFO | LISTTALLY | 49 | 86 | 105 | 131 | 152 | 55 | 95 | 136 | 149 | 145 |
| | FUNCTALLY | 42 | 67 | 69 | 74 | 74 | 46 | 66 | 72 | 73 | 73 |
| STEEP | LISTTALLY | 53 | 69 | 96 | 120 | 134 | 58 | 92 | 109 | 158 | 141 |
| | FUNCTALLY | 43 | 45 | 49 | 53 | 56 | 49 | 62 | 51 | 59 | 52 |
| ULLMAN | LISTTALLY | 91 | 149 | 118 | 157 | 157 | 106 | 157 | 157 | 157 | 157 |
| | FUNCTALLY | 46 | 80 | 95 | 120 | 147 | 56 | 85 | 124 | 170 | 140 |

THE FOLLOWING LIST REPRESENTS THE NUMBER
OF TIMES THE CORRESPONDING METHOD USED THE
LOWEST OR SAME NUMBER OF APPLICATIONS
DURING THE LAST 20 RUNS.

| | ORIG | FIFO | STEEP | ULLMAN |
|---|---|---|---|---|
| LISTTALLY | 5 | 4 | 14 | 0 |
| FUNCTALLY | 0 | 2 | 16 | 2 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE NONLINEAR OPTIMIZING FUNCTION WAS USED

| RUN | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NMNODE | | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 |
| MAXCON | | 2 | 4 | 10 | 20 | 30 | 2 | 4 | 10 | 20 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 10 | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 20 |
| THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS. | | | | | | | | | | | |
| LIFO | LISTTALLY | 21 | 35 | 68 | 92 | 145 | 38 | 100 | 165 | 156 | 187 |
| | FUNCTALLY | 21 | 35 | 68 | 92 | 145 | 38 | 100 | 165 | 156 | 187 |
| FIFO | LISTTALLY | 21 | 32 | 69 | 72 | 113 | 45 | 77 | 116 | 143 | 181 |
| | FUNCTALLY | 20 | 30 | 61 | 57 | 73 | 42 | 73 | 95 | 101 | 99 |
| STEEP | LISTTALLY | 21 | 36 | 69 | 83 | 102 | 38 | 75 | 107 | 124 | 146 |
| | FUNCTALLY | 20 | 34 | 61 | 60 | 54 | 32 | 65 | 74 | 65 | 73 |
| ULLMAN | LISTTALLY | 69 | 49 | 136 | 139 | 197 | 79 | 127 | 193 | 193 | 197 |
| | FUNCTALLY | 20 | 30 | 63 | 81 | 115 | 38 | 68 | 116 | 137 | 165 |

STATISTICAL SUMMARY

STARTSEED FOR THIS RUN = 12345678

THE NONLINEAR OPTIMIZING FUNCTION WAS USED

| RUN | | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NMNODE | | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 |
| MAXCON | | 2 | 4 | 10 | 20 | 2 | 4 | 10 | 20 | 30 | 30 |
| MALGTH | | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| MAXLVL | | 30 | 30 | 30 | 30 | 50 | 50 | 50 | 50 | 30 | 50 |
| THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS. | | | | | | | | | | | |
| LIFO | LISTTALLY | 59 | 90 | 160 | 184 | 78 | 131 | 179 | 209 | 194 | 231 |
| | FUNCTALLY | 59 | 90 | 160 | 184 | 78 | 131 | 179 | 209 | 194 | 231 |
| FIFO | LISTTALLY | 55 | 85 | 137 | 169 | 89 | 137 | 150 | 185 | 174 | 178 |
| | FUNCTALLY | 49 | 72 | 99 | 88 | 77 | 103 | 85 | 96 | 83 | 81 |
| STEEP | LISTTALLY | 52 | 74 | 115 | 142 | 63 | 120 | 129 | 154 | 146 | 173 |
| | FUNCTALLY | 48 | 56 | 60 | 64 | 53 | 82 | 60 | 59 | 57 | 65 |
| ULLMAN | LISTTALLY | 106 | 177 | 197 | 197 | 133 | 193 | 197 | 197 | 197 | 246 |
| | FUNCTALLY | 51 | 93 | 141 | 166 | 67 | 138 | 141 | 188 | 168 | 159 |

THE FOLLOWING LIST REPRESENTS THE NUMBER
OF TIMES THE CORRESPONDING METHOD USED THE
LOWEST OR SAME NUMBER OF APPLICATIONS
DURING THE LAST 20 RUNS.

| | ORIG | FIFO | STEEP | ULLMAN |
|---|---|---|---|---|
| LISTTALLY | 3 | 3 | 17 | 0 |
| FUNCTALLY | 0 | 4 | 18 | 2 |

```
C
C                     C O M P U T E R    P R O G R A M
C
C                                  F O R
C
C                      S I M U L A T I O N    M O D E L
C
C     *****************************************************************
C
C     THIS PROGRAM IS THE FORTRAN IMPLEMENTATION OF THE ALGORITHM Q OUT
C     OF PROF. G. KILDAL ' S
C     GLOBAL EXPRESSION OPTIMISATION DURING COMPILATION    PAGE 40
C      REF. 6
C
C     IN KNUTH NOTATION :
C                             Q1   INITIALIZE
C                             Q2   TERMINATE ?
C                             Q3   SELECT A NODE
C                             Q4   TRAVERSE  ?
C                             Q5   SET STATE
C                             Q5   LOOP
C
C     THE MOTIVATION TO WRITE THIS PROGRAM IS TO TRY SEVERAL
C     MODIFICATION OF THE SELECT STEP TO OPTIMIZE THE TRAVERSE STEP.
C
C     EXPLANATION OF ABREVIATIONS AND NAMES
C
C     NSTART   =  START NODE
C     INTVSM   =  SUM OF INTERVALS UP TO THE CORRESPONDING NODE
C     INTVAL   =  DISTANCE BETWEEN TWO NODES ALONG ONE ARC
C     NGHLST   =   NEIGHBORLIST OR INVESTIGATION LIST " L "
C     NGHBOR   =   NEIGHBOR NODE BEING PROCESSED
C     NEWNGH   =    NODE BEING ENTERED TO THE INVESTIGATION LIST
C
C     ******************************************************************
C
      IMPLICIT INTEGER (O)
      IMPLICIT LOGICAL($)
      DIMENSION NETWRK(100,100,3),RNCONT(400),LEVEL(100),INTVSM(101),
     1   NSMARY(10,20),ITOUCH(101),NANODE(100),IFETCH(101),
     2   MARCON(100),MALGTH(100),MARLVL(100),NORDER(100),ORDLST(100),
     3   NGHLST(2500),IWIN(10)
      COMMON  NETWRK,RNCONT,LEVEL , INTVSM,NGHLST,ISTART,NGHBOR,
     1        NMNODE,NEWNGB ,RNROOT,I, ITOUCH,NSAVE,$NOPRT,NANODE,
     2          MARCON,MALGTH,MARLVL,IDATA ,NORDER,NUMNDE ,
     3          ORDLST,$LIN,IFETCH
C
C     THE FRAME LOOP "IDATA" WILL PROVIDE FOR SEVERAL DISTINCT
```

```
C       PROGRAM GRAPHS. TO AVOID REPETITION OF THE RANDOM NUMBERS
        ISTART = 12345678
        OSTART = ISTART
        $LIN = .TRUE.
        $NOPRT = .TRUE.
6001    DO 6000 LUKA = 1,5
        DO 4 MN = 1,20
        NANODE(MN) = 0
        MARCON(MN) = 0
        MALGTH(MN) = 0
        MARLVL(MN) = 0
4       CONTINUE
        DO 5 NM = 1,10
        IWIN(NM) = 0
          DO 5 MN = 1,20
            NSMARY(NM,MN) = 0
5       CONTINUE
        DO 5500  IDATA = 1 , 20
        CALL DATA
C
C
C       THE FOLLOWING LOOP "ISELCT" WILL PROVIDE FOR THE  CONTROL-
C       VALUE FOR THE COMPUTED GO TO.THAT IS THE SAME GRAPH
C       AND THE SAME START  NODE  ARE USED FOR DIFFERENT ROUTINES
C       SELECT# AND ENTER#
C
        DO 5000  ISELCT= 1,4
C
C       STEP 1 OF THE ALGORITHM  " INITIALIZE"
C       ****************************************************************************
C
        DO 10 KIL1  = 1,101
        INTVSM(KIL1) = 99999
        NGHLST(KIL1) = 0
        ITOUCH(KIL1) = 0
        IFETCH(KIL1) = 0
10      CONTINUE
1       NSTART = 1
        INTVSM(NSTART) = 1
        IF($LIN) INTVSM(1) = 0
        NGHLST(1) = NSTART
        NSAVE = 1
C
C       STEP 2 OF THE ALGORITHM TERMINATE ?
C       ****************************************************************************
C
100     IF (NGHLST(1).EQ.0) GO TO 200
C
```

```
C       STEP 3 OF THE ALGORITHM  SELECT A NODE                            LUK00980
C       ******************************************************************LUK00990
C                                                                         LUK01000
C       SINCE THIS STEP IS THE OBJECT OF FURTHER INVESTIGATION            LUK01010
C       THE FOLLOWING COMPUTED GO TO CONTAINS THE DIFFERENT               LUK01020
C       S E L E C T   R O U T I N E S                                     LUK01030
C                                                                         LUK01040
        GO TO(112,110,115,117),ISELCT                                     LUK01050
110     CALL SELCT1                                                       LUK01060
        GO TO 140                                                         LUK01070
112     CALL SELCT2                                                       LUK01080
        GO TO 140                                                         LUK01090
115     CALL SELCT3                                                       LUK01100
        GO TO 140                                                         LUK01110
117     CALL ULLMAN                                                       LUK01120
        GO TO 200                                                         LUK01130
C                                                                         LUK01140
C       STEP 4/5 OF THE ALGORITHM TRAVERSE AND SET THE STATE              LUK01150
C       ******************************************************************LUK01160
C                                                                         LUK01170
140     IFETCH(NGHBOR) = IFETCH(NGHBOR) +1                                LUK01180
        DO 150 KIL3 = 1,NMNODE                                            LUK01190
        IF(KIL3.EQ.NGHBOR) GO TO 150                                      LUK01200
        IF(NETWRK(NGHBOR,KIL3,1).EQ.0)   GO TO 150                        LUK01210
        NX = NETWRK(NGHBOR,KIL3,1)                                        LUK01220
        NY = NETWRK(NGHBOR,KIL3,2)                                        LUK01230
        NZ = NETWRK(NGHBOR,KIL3,3)                                        LUK01240
C                                                                         LUK01250
C       OPTIMIZING FNCNTION                                               LUK01260
        IF($LIN) INTVAL= INTVSM(NGHBOR) + NX                              LUK01270
        IF(.NOT.$LIN)                                                     LUK01280
       1INTVAL=FLOAT(INTVSM(NGHBOR)**2)/NX**2+FLOAT(INTVSM(NGHBOR))/NY+NZ LUK01290
C                                                                         LUK01300
C                                                                         LUK01310
C       MEETFUNCTION                                                      LUK01320
        IF(INTVAL.GE.INTVSM(KIL3))GO TO 150                               LUK01330
C                                                                         LUK01340
        INTVSM(KIL3)=INTVAL                                               LUK01350
        NEWNGB = KIL3                                                     LUK01360
        ITOUCH(NEWNGB) = ITOUCH(NEWNGB) +1                                LUK01370
C                                                                         LUK01380
C       THE FOLLOWING COMPUTED GO TO CONTAINS THE DIFFERENT               LUK01390
C       E N T E R   R O U T I N E S                                       LUK01400
C                                                                         LUK01410
        GO TO(113,111,116,118),ISELCT                                     LUK01420
111     CALL ENTER1                                                       LUK01430
        GO TO 150                                                         LUK01440
113     CALL ENTER3                                                       LUK01450
```

```
      GO TO 150                                                         LUK01460
116   CALL ENTER1                                                       LUK01470
      GO TO 150                                                         LUK01480
118   GO TO 200                                                         LUK01490
C                                                                       LUK01500
150   CONTINUE                                                          LUK01510
      GO TO 100                                                         LUK01520
C                                                                       LUK01530
C     ******************************************************************LUK01540
C     FINAL STEP TO GET THE RESULT AND THE STATISTICS                   LUK01550
C                                                                       LUK01560
200   ISUM = 1                                                          LUK01570
      LSUM = 0                                                          LUK01580
      DO 60 I = 1,100                                                   LUK01590
      ISUM = ISUM + ITOUCH(I)                                           LUK01600
      LSUM = LSUM + IFETCH(I)                                           LUK01610
60    CONTINUE                                                          LUK01620
      IF($NOPRT) GO TO 4900                                             LUK01630
      WRITE(6,1000) IDATA, ISELCT                                       LUK01640
      WRITE(6,1001)(I,I=1,20)                                           LUK01650
      WRITE(6,1002)(ITOUCH(I),I=1,20)                                   LUK01660
      WRITE(6,1006) ISUM                                                LUK01670
      WRITE(6,1017)(IFETCH(I),I = 1,20)                                 LUK01680
      WRITE(6,1018) LSUM                                                LUK01690
      DO 4800 I = 1,NMNODE                                              LUK01700
      WRITE(6,1016) I , INTVSM(I)                                       LUK01710
4800  CONTINUE                                                          LUK01720
4900  NSMARY(ISELCT,IDATA) = ISUM                                       LUK01730
      NSMARY(ISELCT + 5,IDATA) = LSUM                                   LUK01740
5000  CONTINUE                                                          LUK01750
      LSTVAL = MINO(NSMARY(1,IDATA),NSMARY(2,IDATA),NSMARY(3,IDATA) ,   LUK01760
     1NSMARY(4,IDATA)   )                                               LUK01770
      LFUVAL = MINO(NSMARY(6,IDATA),NSMARY(7,IDATA),NSMARY(8,IDATA)  ,  LUK01780
     1NSMARY(9,IDATA)   )                                               LUK01790
      DO 4950 LS = 1,4                                                  LUK01800
      IF(LSTVAL.EQ.NSMARY(LS,IDATA))IWIN(LS) = IWIN(LS) + 1             LUK01810
      IF(LFUVAL.EQ.NSMARY(LS+5,IDATA))IWIN(LS+5)=IWIN(LS+5) +1          LUK01820
4950  CONTINUE                                                          LUK01830
5500  CONTINUE                                                          LUK01840
      IRT = 1                                                           LUK01850
      I1 = 1                                                            LUK01860
      I2 = 10                                                           LUK01870
5501  WRITE(6,1005)                                                     LUK01880
      WRITE(6,1020) OSTART                                              LUK01890
      IF($LIN) WRITE(6,1021)                                            LUK01900
      IF(.NOT.$LIN) WRITE(6,1022)                                       LUK01910
      WRITE(6,1007)(I,I=I1,I2)                                          LUK01920
      WRITE(6,1011)(NANODE(I),I=I1,I2)                                  LUK01930
```

```
      WRITE(6,1012)(MARCON(I),I=I1,I2)
      WRITE(6,1013)(MALGTH(I),I=I1,I2)
      WRITE(6,1014)(MARLVL(I),I=I1,I2)
      WRITE(6,1028)
      WRITE(6,1008)(NSMARY(1,I),I =I1,I2)
      WRITE(6,1019)(NSMARY(6,I),I = I1,I2)
      WRITE(6,1009)(NSMARY(2,I),I =I1,I2)
      WRITE(6,1019)(NSMARY(7,I),I = I1,I2)
      WRITE(6,1010)(NSMARY(3,I),I =I1,I2)
      WRITE(6,1019)(NSMARY(8,I),I = I1,I2)
      WRITE(6,1015)(NSMARY(4,I),I=I1,I2)
      WRITE(6,1019)(NSMARY(9,I),I =I1,I2)
      IF(IRT.EQ.2) GO TO 5502
      IRT = 2
      I1 = 11
      I2 = 20
      GO TO 5501
 5502 WRITE(6,1023)
      WRITE(6,1027)
      WRITE(6,1029)
      WRITE(6,1030)
      WRITE(6,1024)
      WRITE(6,1025)IWIN(1),IWIN(2),IWIN(3),IWIN(4)
      WRITE(6,1026)IWIN(6),IWIN(7),IWIN(8),IWIN(9)
 1000 FORMAT('0',35X,'STATISTICS FOR THE GRAPH ',I4,4X,'ROUTINE',I4////)
 1001 FORMAT('0','NODES       ',20I4)
 1002 FORMAT('0','LISTTALLY ',20I4)
 1005 FORMAT('1',////////,16X,'STATISTICAL SUMMARY ')
 1006 FORMAT('0','THE TOTAL NUMBER OF NODES TRAVERSED IS ',I6/   )
 1007 FORMAT('0',15X,'RUN       ',11X,10I4)
 1008 FORMAT('0',15X,'LIFO     LISTTALLY ',10I4)
 1009 FORMAT('0',15X,'FIFO     LISTTALLY ',10I4)
 1010 FORMAT('0',15X,'STEEP    LISTTALLY ',10I4)
 1011 FORMAT('0',15X,'NMNODE   ',10X,10I4)
 1012 FORMAT('0',15X,'MAXCON   ',10X,10I4)
 1013 FORMAT('0',15X,'MALGTH   ',10X,10I4)
 1014 FORMAT('0',15X,'MAXLVL   ',10X,10I4)
 1015 FORMAT('0',15X,'ULLMAN  LISTTALLY ',10I4)
 1016 FORMAT(' ','NODE ',I4,'   INTERVALSUM',I10)
 1017 FORMAT(' ',15X,'FUNCTALLY ',10I4)
 1018 FORMAT('0','TOTAL NUMBER OF FUNCTION APPLICATION ',I4)
 1019 FORMAT('0',15X,8X,'FUNCTALLY ',10I4)
 1020 FORMAT('0',15X,'STARTSEED FOR THIS  RUN = ',I10)
 1021 FORMAT('0',15X,'THE LINEAR OPTIMIZING FUNCTION WAS USED ')
 1022 FORMAT('0',15X,'THE NONLINEAR OPTIMIZING FUNCTION WAS USED')
 1023 FORMAT(////,'0',15X,'THE FOLLOWING LIST REPRESENTS THE NUMBER')
 1024 FORMAT('0',30X,'     ORIG     FIFO    STEEP   ULLMAN')
 1025 FORMAT('0',15X,'LISTTALLY',10X,I4,4X,I4,4X,I4,4X,I4)
```

```
1026  FORMAT('0',15X,'FUNCTALLY',10X,I4,4X,I4,4X,I4,4X,I4)              LUK02420
1027  FORMAT('0',15X,'OF TIMES THE CORRESPONDING METHOD USED THE    ')   LUK02430
1028  FORMAT('0',15X,'THE FOLLOWING NUMBERS ARE THE COUNTER CONTENTS.') LUK02440
1029  FORMAT('0',15X,'LOWEST  OR SAME NUMBER OF APPLICATIONS')          LUK02450
1030  FORMAT('0',15X,'DURING THE LAST 20 RUNS.')                        LUK02460
6000  CONTINUE                                                           LUK02470
      IF(.NOT.$LIN) GO TO 6002                                           LUK02480
      $LIN = .FALSE.                                                     LUK02490
      GO TO 6001                                                         LUK02500
6002  WRITE(6,6003)                                                      LUK02510
6003   FORMAT('1','  ')                                                  LUK02520
       STOP                                                              LUK02530
      END                                                                LUK02540
```

```
      SUBROUTINE SELCT1
C
C     THIS PROCEDURE SELECTS THE FIRST ONE OF
C     THE LIST TO BE THE NEXT SELECTED NODE. THEN ALL OTHERS ARE
C     CHANGED TO THE LOWER ARRAYBOX
C
      IMPLICIT INTEGER (O)
      IMPLICIT LOGICAL($)
      DIMENSION NETWRK(100,100,3),RNCONT(400),LEVEL(100),INTVSM(101),
     1   NSMARY(10,20),ITOUCH(101),NANODE(100),IFETCH(101),
     2   MARCON(100),MALGTH(100),MARLVL(100),NORDER(100),ORDLST(100),
     3   NGHLST(2500)
      COMMON  NETWRK,RNCONT,LEVEL  , INTVSM,NGHLST,ISTART,NGHBOR,
     1        NMNODE,NEWNGB  ,RNROOT,I,  ITOUCH,NSAVE,$NOPRT,NANODE,
     2          MARCON,MALGTH,MARLVL,IDATA ,NORDER,NUMNDE ,
     3          ORDLST,$LIN,IFETCH
      NGHBOR = NGHLST(1)
      IF($NOPRT) GO TO 9
      WRITE(6,1000) NGHBOR
1000  FORMAT(' ','SELECTED ',I6)
9     DO 10 ISEL =1,NMNODE
      NGHLST(ISEL) = NGHLST(ISEL + 1)
      IF(NGHLST(ISEL).EQ.0) RETURN
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SELCT2
C
C     THIS ROUTINE WILL SELECT THAT NODE WHICH WAS LAST ENTERED
C
      IMPLICIT INTEGER (O)
      IMPLICIT LOGICAL($)
      DIMENSION NETWRK(100,100,3),RNCONT(400),LEVEL(100),INTVSM(101),
     1   NSMARY(10,20),ITOUCH(101),NANODE(100),IFETCH(101),
     2   MARCON(100),MALGTH(100),MARLVL(100),NORDER(100),ORDLST(100),
     3   NGHLST(2500)
      COMMON  NETWRK,RNCONT,LEVEL , INTVSM,NGHLST,ISTART,NGHBOR,
     1        NMNODE,NEWNGB ,RNROOT,I, ITOUCH,NSAVE,$NOPRT,NANODE,
     2          MARCON,MALGTH,MARLVL,IDATA ,NORDER,NUMNDE    ,
     3          ORDLST,$LIN,IFETCH
C
      NGHBOR = NGHLST(NSAVE)
      IF($NOPRT) GO TO 9
      WRITE(6,1000) NGHBOR
1000  FORMAT(' ','SELECTED ',I6)
9     NGHLST(NSAVE) = 0
      NSAVE = NSAVE -1
      IF(NSAVE.EQ.0)NSAVE = 1
      RETURN
      END
```

```
      SUBROUTINE SELCT3                                                 LUK03110
C                                                                       LUK03120
C     THIS ROUTINE WILL SELECT THE NODE WITH THE LOWEST INTERVALSUM     LUK03130
C                                                                       LUK03140
      IMPLICIT INTEGER (O)                                              LUK03150
      IMPLICIT LOGICAL($)                                               LUK03160
      DIMENSION NETWRK(100,100,3),RNCONT(400),LEVEL(100),INTVSM(101),   LUK03170
     1   NSMARY(10,20),ITOUCH(101),NANODE(100),IFETCH(101),             LUK03180
     2   MARCON(100),MALGTH(100),MARLVL(100),NORDER(100),ORDLST(100),   LUK03190
     3   NGHLST(2500)                                                   LUK03200
      COMMON  NETWRK,RNCONT,LEVEL  , INTVSM,NGHLST,ISTART,NGHBOR,       LUK03210
     1        NMNODE,NEWNGB  ,RNROOT,I,  ITOUCH,NSAVE,$NOPRT,NANODE,    LUK03220
     2           MARCON,MALGTH,MARLVL,IDATA ,NORDER,NUMNDE   ,          LUK03230
     3           ORDLST,$LIN,IFETCH                                     LUK03240
      ISAVE = 1                                                         LUK03250
      NGHBOR = NGHLST(1)                                                LUK03260
      DO 10 ISORT = 1,NMNODE                                            LUK03270
      NEXT = NGHLST(ISORT +1)                                           LUK03280
      IF(NEXT.EQ.0) GO TO 15                                            LUK03290
      IF(INTVSM(NGHBOR).LE.INTVSM(NEXT)) GO TO 10                       LUK03300
      ISAVE = ISORT +1                                                  LUK03310
      NGHBOR=NEXT                                                       LUK03320
10    CONTINUE                                                          LUK03330
15    IF($NOPRT) GO TO 9                                                LUK03340
      WRITE(6,1000) NGHBOR                                              LUK03350
1000  FORMAT(' ','SELECTED ',I6)                                        LUK03360
9     DO 20 ICHGE = ISAVE,NMNODE                                        LUK03370
      NGHLST(ICHGE) = NGHLST(ICHGE + 1)                                 LUK03380
      IF(NGHLST(ICHGE).EQ.0) RETURN                                     LUK03390
20    CONTINUE                                                          LUK03400
      RETURN                                                            LUK03410
      END                                                               LUK03420
```

```
      SUBROUTINE ENTER1
C
C     THIS PROCEDURE WILL ADD THE NEW NEIGHBOR TO THE END OF THE ARRAY
C     IF IT IS NOT ALREADY IN THE ARRAY
C     FOR STATISTIC REASONS WRITE THE NEIGHLIST AND THE NODE,
C     WHICH IS TO BE ENTERED
C
      IMPLICIT INTEGER (O)
      IMPLICIT LOGICAL($)
      DIMENSION NETWRK(100,100,3),RNCONT(400),LEVEL(100),INTVSM(101),
     1   NSMARY(10,20),ITOUCH(101),NANODE(100),IFETCH(101),
     2   MARCON(100),MALGTH(100),MARLVL(100),NORDER(100),ORDLST(100),
     3   NGHLST(2500)
      COMMON  NETWRK,RNCONT,LEVEL , INTVSM,NGHLST,ISTART,NGHBOR,
     1        NMNODE,NEWNGB ,RNROOT,I, ITOUCH,NSAVE,$NOPRT,NANODE,
     2          MARCON,MALGTH,MARLVL,IDATA ,NORDER,NUMNDE      ,
     3          ORDLST,$LIN,IFETCH
C
C
      DO 10  IENT = 1,NMNODE
      IF(NGHLST(IENT).EQ.0) NGHLST(IENT) = NEWNGB
      IF(NGHLST(IENT).EQ.NEWNGB) GO TO 20
10    CONTINUE
20    IF($NOPRT) GO TO 9
      WRITE(6,1000)(NGHLST(I),I=1,20)
1000  FORMAT(' ','L = ',20I4)
9     RETURN
      END
```

```
      SUBROUTINE ENTER3
C
C     THIS ROUTINE WILL BUILD A STACK OF NODES FOR THE SELECT STEP
C
      IMPLICIT INTEGER (O)
      IMPLICIT LOGICAL($)
      DIMENSION NETWRK(100,100,3),RNCONT(400),LEVEL(100),INTVSM(101),
     1   NSMARY(10,20),ITOUCH(101),NANODE(100),IFETCH(101),
     2   MARCON(100),MALGTH(100),MARLVL(100),NORDER(100),ORDLST(100),
     3   NGHLST(2500)
      COMMON  NETWRK,RNCONT,LEVEL , INTVSM,NGHLST,ISTART,NGHBCR,
     1        NMNODE,NEWNGB ,RNROOT,I,  ITOUCH,NSAVE,$NOPRT,NANODE,
     2           MARCON,MALGTH,MARLVL,IDATA ,NORDER,NUMNDE  ,
     3           ORDLST,$LIN,IFETCH
      IF(NSAVE.NE.1)GO TO 10
      IF(NGHLST(NSAVE).NE.0) GO TO 10
      NSAVE = 0
10    NSAVE = NSAVE +1
      NGHLST(NSAVE) = NEWNGB
      IF($NOPRT) GO TO 9
      WRITE(6,1000)(NGHLST(I),I=1,20)
1000  FORMAT(' ','L = ',20I4)
9     RETURN
      END
```

```
      SUBROUTINE DATA
C
C     THIS PROGRAM CONSTRUCTS A PROGRAM FLOW GRAPH IN THE FOLLOWING
C     WAY;
C     1  START WITH THE NODE "1" AS THE ENTRY.
C     2  GENERATE FOR EACH OUTGOING ARC THE INGOING NODE AND THE
C        ARCLENGTH. IN THE CASE OF STRICTLY PARALLEL ARCS ADD THE
C        DISTANCES AND TAKE ONE ARC ONLY.
C     3  TO AVOID A DISCONNECTED GRAPH PICK BY LLRANDOM ONE OF THE
C        INGOING NODES  OF THE LAST LEVEL AS THE NEW ROOT FOR THE
C        NEXT LEVEL.
C     4  GO TO 2
C
C     THE OUTPUT OF THIS ROUTINE IS THE GRAPH IN MATRIX FORM AND THE
C     INGOING NODES OF THE LAST LEVEL, AVAILABLE IN COMMON
C     TO SIMULATE DIFFERENT GRAPHS CHANGE:
C            THE MAXIMAL NUMBER OF ARCS
C            THE MAXIMAL NUMBER OF LEVELS
C            THE MAXIMAL NUMBER OF LENGTH BETWEEN THE NODES
C
      IMPLICIT INTEGER (O)
      IMPLICIT LOGICAL($)
      DIMENSION NETWRK(100,100,3),RNCONT(400),LEVEL(100),INTVSM(101),
     1   NSMARY(10,20),ITOUCH(101),NANODE(100),IFETCH(101),
     2   MARCON(100),MALGTH(100),MARLVL(100),NORDER(100),ORDLST(100),
     3   NGHLST(2500)
      COMMON  NETWRK,RNCONT,LEVEL  , INTVSM,NGHLST,ISTART,NGHBOR,
     1        NMNODE,NEWNGB  ,RNROOT,I,  ITOUCH,NSAVE,$NOPRT,NANODE,
     2          MARCON,MALGTH,MARLVL,IDATA ,NORDER,NUMNDE,
     3          ORDLST,$LIN,IFETCH
C*****************************************************************
C
C     DECLARATION OF VARIABLES
C
C     NMNODE = NUMBER OF NODES IN THE NETWORK  MAX . 60
C     MAXCON = MAXIMAL NUMBER OF OUTGOING ARCS.
C                 LIMITED TO 60 ARCS DUE TO RANDOMNUMBERARRAY
C     RNCONT = RANDOMNUMBER CONTAINER
C     ISTART = STARTVALUE FOR LLRANDOM
C     NMCON  = NUMBER OF CONNECTIONS
C     RAMCON = CONTAINER FOR RANDOMNUMBERS
C     MXLGTH = MAXIMAL LENGTH OF THE INTERVALL
C     NETWRK =  MATRIX  FOR THE GENERATED INTERVALLLENGTH
C     RNROOT = RANDOM ROOT FOR THE NEXT LEVEL
C     MAXLVL = MAXIMAL NUMBER OF LEVELS
C     LEVEL = ARRAY FOR THE  END NODES OF THE LEVEL
C     LVLCNT = LEVELCOUNTER
C*****************************************************************
```

```
C
C
C     NECESSARY FOR THE USED RANDOMNUMBERGENERATOR LLRANDOM
      CALL OVFLOW
C
C     ZERO OUT THE NETWORK MATRIX
C
10    DO 100  I1 = 1, 100
      LEVEL(I1) = 0
       DO 100  I2 = 1,100
             DO 100 I3 = 1,3
         NETWRK(I1,I2,I3)  =  0
100   CONTINUE
C
      READ(5,1000) NMNODE, MAXCON,  MXLGTH,MAXLVL
      NANODE(IDATA) = NMNODE
      MARCON(IDATA) = MAXCON
      MALGTH(IDATA) = MXLGTH
      MARLVL(IDATA) = MAXLVL
C
C     *****************************************************************
C
C     GENERATE THE LEVELS
C
101   LEVEL(1) = 1
      NMCON = 1
      DO 200 LVLCNT = 1,MAXLVL
      CALL RANDOM(ISTART,RNROOT,1)
      NPOINT = RNROOT  * NMCON + 0.5
      IF(NPOINT.EQ.0)NPOINT = 1
      NDEOUT = LEVEL(NPOINT)
      NMCON = MAXCON
C
C     TO DETERMINE THE LENGTH OF THE RNUMBER ARRAY  TAKE
C     FOUR TIMES THE NUMBER OF CONNECTIONS.
      NMCNX2  = 4 * NMCON
C
C     GENERATE THE OUTGOING  ARCS AND THEIR LENGTH
C
         CALL RANDOM(ISTART,RNCONT,NMCNX2)
      DO 225 ICLEAR = 1,100
      LEVEL(ICLEAR) = 0
225   CONTINUE
      I = 0
         DO 250 ICOUNT  = 1,NMCNX2,4
           NDEIN = RNCONT(ICOUNT) * NMNODE  + 0.5
           IF ( NDEIN.EQ.0.OR.NDEIN.EQ.NDEOUT)NDEIN =(NDEIN+3)/2
              DO 260 NJ =1,3
```

```
          INTVAL  = RNCONT(ICOUNT + NJ) * MXLGTH  +  0.5
      IF (INTVAL.EQ.0) INTVAL = 1
              NETWRK(NDEOUT,NDEIN,NJ) = NETWRK(NDEOUT,NDEIN,NJ)+INTVAL
260   CONTINUE
      I = I +1
      LEVEL(I) = NDEIN
250   CONTINUE
200   CONTINUE
C
C     ******************************************************************
C
C     PRINT THE NETWORK IN MATRIX FORM. IF THE NUMBER OF NODES
C     IS LESS THAN THE PREDEFINED ONE THE LAST COLLUMNS ARE
C     FILLED WITH ZEROS
      IF($NOPRT) GO TO 615
      DO 610 NJ = 1,3
      WRITE(6,1010)( I3, I3 = 1,20)
      DO 600 I4 = 1,NMNODE
       WRITE(6,1011) I4,(NETWRK(I4,I5,NJ), I5 = 1,20)
600   CONTINUE
610   CONTINUE
C     ******************************************************************
1000  FORMAT(4I4)
1010  FORMAT('0','NODES    ',20I4/)
1011  FORMAT(' ',2X,I4,2X,20I4)
615   RETURN
      END
```

```
      SUBROUTINE ULLMAN                                                 LUK05260
C                                                                       LUK05270
C     THIS ROUTINE IS THE IMPLEMENTATTION OF   THE                      LUK05280
C     DEPTH FIRST ALGORITHM D                                           LUK05290
C                                                                       LUK05300
      IMPLICIT INTEGER (O)                                              LUK05310
      IMPLICIT LOGICAL($)                                               LUK05320
      DIMENSION NETWRK(100,100,3),RNCONT(400),LEVEL(100),INTVSM(101),   LUK05330
     1   NSMARY(10,20),ITOUCH(101),NANODE(100),IFETCH(101),             LUK05340
     2   MARCON(100),MALGTH(100),MARLVL(100),NORDER(100),ORDLST(100),   LUK05350
     3   NGHLST(2500)                                                   LUK05360
      COMMON  NETWRK,RNCONT,LEVEL  , INTVSM,NGHLST,ISTART,NGHBOR,       LUK05370
     1        NMNCDE,NEWNGB  ,RNROOT,I,  ITOUCH,NSAVE,$NOPRT,NANODE,    LUK05380
     2           MARCON,MALGTH,MARLVL,IDATA ,NORDER,NUMNDE   ,          LUK05390
     3           ORDLST,$LIN,IFETCH                                     LUK05400
C                                                                       LUK05410
      CALL RECRDR                                                       LUK05420
      IF ($NOPRT) GO TO 5                                               LUK05430
      DO 6 I = 1,NMNCDE                                                 LUK05440
      WRITE(6,1010) I,NORDER(I)                                         LUK05450
1010  FORMAT(' ','THEORDER OF NODE',I4,'  IS',I4)                       LUK05460
6     CONTINUE                                                          LUK05470
C                                                                       LUK05480
C     STEP ONE                                                          LUK05490
C                                                                       LUK05500
 5    DO 10 J = 2,NUMNDE                                                LUK05510
      NSON = ORDLST(J)                                                  LUK05520
      ITOUCH(NSON) = ITOUCH(NSON) + 1                                   LUK05530
      DO 20 JPRE = 1,NMNODE                                             LUK05540
      IF(JPRE.EQ.NSON) GO TO 20                                         LUK05550
      IF(NETWRK(JPRE,NSON,1).EQ.0) GO TO 20                             LUK05560
      IF(NORDER(JPRE).GT.J) GO TO 20                                    LUK05570
      NX = NETWRK(JPRE,NSON,1)                                          LUK05580
      NY = NETWRK(JPRE,NSON,2)                                          LUK05590
      NZ = NETWRK(JPRE,NSON,3)                                          LUK05600
C                                                                       LUK05610
C     OPTIMIZING FNCNTION                                               LUK05620
      IF($LIN) INTVAL = INTVSM(JPRE) + NX                               LUK05630
      IF(.NOT.$LIN)                                                     LUK05640
     1INTVAL=FLOAT(INTVSM(JPRE)**2)/NX**2+FLOAT(INTVSM(JPRE))/NY + NZ   LUK05650
C                                                                       LUK05660
C                                                                       LUK05670
C     MEETFUNCTION                                                      LUK05680
      IF(INTVSM(NSON).LE.INTVAL)GO TO 20                                LUK05690
C                                                                       LUK05700
      INTVSM(NSON) = INTVAL                                             LUK05710
      IFETCH(NSON) = IFETCH(NSON) +1                                    LUK05720
20    CONTINUE                                                          LUK05730
```

```
10    CCNTINUE                                                           LUK05740
C                                                                        LUK05750
C                                                                        LUK05760
C                                                                        LUK05770
C     STEP  TWO                                                          LUK05780
C                                                                        LUK05790
      $CHNGE = .TRUE.                                                    LUK05800
      NULCNT = 1                                                         LUK05810
30    IF($CHNGE) GO TO 40                                                LUK05820
      IF(.NOT.$NOPRT) WRITE(6,1000) NULCNT                               LUK05830
1000  FORMAT('0','THE NUMBER OF ITERERATION IS ',I6)                     LUK05840
      RETURN                                                             LUK05850
40    $CHNGE=.FALSE.                                                     LUK05860
      NULCNT = NULCNT + 1                                                LUK05870
      DO 50 J = 2,NUMNDE                                                 LUK05880
      NSON = ORDLST(J)                                                   LUK05890
      ITOUCH(NSON) = ITOUCH(NSON) + 1                                    LUK05900
      DO 60 JPRE = 1,NMNODE                                              LUK05910
      IF(JPRE.EQ.NSON) GO TO 60                                          LUK05920
      IF(NETWRK(JPRE,NSON,1).EQ.0) GO TO 60                              LUK05930
      NX = NETWRK(JPRE,NSON,1)                                           LUK05940
      NY = NETWRK(JPRE,NSON,2)                                           LUK05950
      NZ = NETWRK(JPRE,NSON,3)                                           LUK05960
C                                                                        LUK05970
C     OPTIMIZING FNCNTION                                                LUK05980
      IF($LIN) INTVAL = INTVSM(JPRE) + NX                                LUK05990
      IF(.NOT.$LIN)                                                      LUK06000
     1INTVAL=FLOAT(INTVSM(JPRE)**2)/NX**2+FLOAT(INTVSM(JPRE))/NY + NZ    LUK06010
C                                                                        LUK06020
C                                                                        LUK06030
C     MEETFUNCTION                                                       LUK06040
      IF(INTVSM(NSON).LE.INTVAL)GO TO 60                                 LUK06050
C                                                                        LUK06060
      INTVSM(NSON) = INTVAL                                              LUK06070
      IFETCH(NSON) = IFETCH(NSON) +1                                     LUK06080
      $CHNGE = .TRUE.                                                    LUK06090
60    CCNTINUE                                                           LUK06100
50    CONTINUE                                                           LUK06110
      GO TO 30                                                           LUK06120
      END                                                                LUK06130
```

```
      SUBROUTINE REORDR
C
C     THIS ROUTINE WILL DETERMINE THE ORDER OF THE NODES           L
C     DUE TO THE ALGORITHM D FOR DFST FLOWGRAPH
      IMPLICIT INTEGER (O)
      IMPLICIT LOGICAL($)
      DIMENSION NETWRK(100,100,3),RNCONT(400),LEVEL(100),INTVSM(101),
     1   NSMARY(10,20),ITOUCH(101),NANODE(100),IFETCH(101),
     2   MARCON(100),MALGTH(100),MARLVL(100),NORDER(100),ORDLST(100),
     3   NGHLST(2500)
      COMMON   NETWRK,RNCONT,LEVEL , INTVSM,NGHLST,ISTART,NGHBOR,
     1         NMNODE,NEWNGB ,RNROOT,I, ITOUCH,NSAVE,$NOPRT,NANODE,
     2           MARCON,MALGTH,MARLVL,IDATA ,NORDER,NUMNDE  ,
     3           ORDLST,$LIN,IFETCH
C     ZERO OUT THE ORDER ATRIBUTE
        DO 10 IORD = 1,NMNODE
       NORDER(IORD) = 0
      ORDLST(IORD) = 0
10     CONTINUE
      NORDER(1) = 1
C      DETERMINE THE NUMBER OF NODES IN THIS NETWORK
      NUMNDE = 0
       DO 20 NUM1 = 1,NMNODE
       NSUM = 0
       DO 30 NUM2 = 1,NMNODE
      NSUM = NSUM + NETWRK(NUM1,NUM2,1)+NETWRK(NUM2,NUM1,1)
30     CONTINUE
       IF(NSUM.NE.0)NUMNDE = NUMNDE + 1
20     CONTINUE
      IN= NUMNDE
       NSAVE =1
      NPRED = 1
60        NMEMO = NSAVE
       DO 40 IPLUS = 1, NMNODE
C      TO GET THE RIGHTMOST SON
      IMIN = NMNODE - IPLUS +1
       IF(NETWRK(NPRED,IMIN,1).EQ.0) GO TO 40
        IF(NORDER(IMIN).GT.0) GO TO 40
       IF(NORDER(IMIN).LT.0) GO TO 40
       NORDER(IMIN) =-10
       NEWNGB = IMIN
       CALL ENTER3
       NPRED = NGHLST(NSAVE)
       GO TO 60
40        CONTINUE
       NORDER(NPRED) = NUMNDE
      ORDLST(NUMNDE) = NPRED
      NUMNDE = NUMNDE - 1
```

```
      NSAVE = NSAVE -1                                                  LUK06640
      IF(NPRED.EQ.1) GO TO 70                                           LUK06650
      NPRED = NGHLST(NSAVE)                                             LUK06660
      GO TO 60                                                          LUK06670
70       IF(NUMNDE.EQ.0) GO TO 80                                       LUK06680
      WRITE(6,1000)                                                     LUK06690
1000   FORMAT('  EFROR IN REORDERING     ')                             LUK06700
      RETURN                                                            LUK06710
80    NUMNDE = IN                                                       LUK06720
      RETURN                                                            LUK06730
      END                                                               LUK06740
```

# BIBLIOGRAPHY


1. Busacker, Robert G., and Saaty, Thomas L., Finite Graphs and Networks: An Introduction With Application, Mc Graw-Hill Inc., 1965.

2. Hecht, M.S. and Ullman, J.D., Analysis of a Simple Algorithm for Global Data Flow Problems, Proceedings of ACM Conference Boston, Mass., October 1973.

3. Kam, J. B., and Ullman, J. D., Global Optimization Problems and Iterative Algorithms, Technical Report Number 146 January, 1974 Princeton University.

4. Kildall, G. A., Algol-E An Experimental Approach to the Study of Programming Languages., NPS -53-KI2011A Navy Postgr.School June 1972 .

5. Kildall, G. A., Global Expression Optimization During Compilation, TR 72-06-02 Computer Science Group University of Washington June 1972.

6. Kildall, G. A., A Unified Approach to Global Program Optimization, Proceedings of ACM Conference Boston, Mass., October 1973 .

7. Learmonth, G. P., and Lewis, P.A.W., Random Number Generator Package LLRandom., NPS 55LW 73061A Navy Postgr. School June 1973 .

INITIAL DISTRIBUTION LIST

| | No. Copies |
|---|---|
| Library, Code 0212<br>Naval Postgraduate School<br>Monterey, California 93940 | 2 |
| Department Chairman, Code 72<br>Computer Science Group<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| Asst. Professor Gary A. Kildall,<br>Code 72Kd (advisor)<br>Computer Science Group<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| Asoc. Professor Uno Kodres,<br>Code 72Kr (second reader)<br>Computer Science Group<br>Naval Postgraduate School<br>Monterey, California 93940 | 1 |
| LCDR. Norbert Lukasczyk, FGN<br>(student)<br>23 Kiel<br>Pestalozzistr. 97<br>Federal Republic of Germany | 1 |
| Marineamt -A1-<br>294 Wilhelmshaven<br>Federal Republic of Germany | 1 |
| Dokumentationszentrale der Bundeswehr (See)<br>53 Bonn<br>Friedrich Ebert Allee 34<br>Federal Republic of Germany | 1 |
| Defense Documentation Center<br>Cameron Station<br>Alexandria, Virginia 22314 | 2 |

152425
Thesis
L8928 Lukasczyk
c.1 An investigation of selection methods for a simple program flow analysis algorithm.

152425
Thesis
L8928 Lukasczyk
c.1 An investigation of selection methods for a simple program flow analysis algorithm.

thesL8928
An investigation of selection methods fo
3 2768 002 12397 8
DUDLEY KNOX LIBRARY